GYMNASE DE RENENS

TRAVAIL DE MATURITÉ 2021

Simulation informatique d'un trébuchet en JavaScript

Répondant
Vincent VUILLE

Adrien BARMAZ
Adrien MARTELLI
Simon PIGUET

Renens, novembre 2021

Simulation informatique d'un trébuchet en JavaScript

Modélisation en ligne d'un trébuchet à contrepoids

Trébuchet du château des Roure, en Ardèche [38]

www.trebuket.org

Adrien BARMAZ
Adrien MARTELLI
Simon PIGUET

Résumé

Le projet consiste en la simulation 2D d'un trébuchet en utilisant le langage de programmation JavaScript. L'objectif final étant de pouvoir paramétrer les caractéristiques physiques de l'engin et d'observer, en conséquence, une modification du mouvement. Pour ceci, nous utilisons les équations de Lagrange, des outils d'intégration numérique ainsi que les lois du mouvement énoncées par Newton.

Remerciements

Nous souhaitons remercier M. Julien Besson pour les réponses apportées à nos interrogations, notamment sur la dynamique des fluides. Nous souhaitons ensuite adresser nos remerciements à Mme Georgina Kohler Barmaz pour la relecture et la correction attentive de nos erreurs de langue. Nous saluons également notre collègue Piotr Maleika, qui nous a volontiers partagé des données relatives à son modèle réduit. Nous remercions également Mme Alice Bovey pour la relecture attentive et son aide au formatage du rapport. Pour finir, nous voulons remercier M. Vincent Vuille pour son rôle de répondant, son soutien et ses conseils.

Table des matières

Introduction

> Est-ce qu'on peut s'en servir pour donner de l'élan à un pigeon ?
>
> Yvain, Chevalier au Lion [8]

Le trébuchet est une arme de siège, originaire de Chine, qui s'est imposée en Europe occidentale à la fin du Moyen Âge. Son mécanisme reprend le principe de la fronde et exploite la force gravitationnelle d'un contrepoids, aussi appelé "huche", afin de propulser des boulets de pierre. Malgré la faible fréquence de tir de l'engin, maximum deux par heure [11], son utilisation permet de détruire les plus épaisses fortifications de pierre. L'objet peut mesurer une vingtaine de mètres et lancer des charges de cent kilogrammes ; quant au contrepoids, il peut varier entre dix et vingt tonnes [37][38].

Définition des attentes

Nous souhaitons simuler le mouvement du trébuchet lors du lancer ainsi que la trajectoire balistique du projectile jusqu'à ce qu'il touche le sol. Nous devons pouvoir modifier les caractéristiques physiques du modèle (longueur des bras, de la fronde, masse du contrepoids, vitesse du vent, etc.). La simulation doit être correcte et vraisemblable, de plus, le dessein final étant de proposer la simulation sur un site internet, l'interface doit être facile d'utilisation, esthétique ainsi que fluide. Un bon exemple de nos attentes est le site `virtualtrebuchet.com` [41].

Sources

Au fur et à mesure du travail, nous avons accumulé une quarantaine de sources sur le sujet, la plupart concernant la mécanique Lagrangienne et l'intégration numérique. Nous nous sommes particulièrement basés sur un travail de Bachelor provenant de l'Université de technologie de Delft, aux Pays-Bas. L'auteur de ce rapport se nomme Robin de Jong et il a été supervisé par Dr. B.J. Meulenbroek. Ce travail a été publié en janvier 2020 [21]. Afin de vérifier notre modèle, nous avons également étudié un travail provenant de l'Université d'Etat de Weber, en Utah. L'auteur en est Donald B. Siano, son travail a été publié en mars 2001 [33].

Choix du langage de programmation

La question du langage de programmation s'est posée très tôt dans notre travail, en effet, nous avons longuement hésité entre Python et JavaScript.
Python est plus pratique pour effectuer de longs calculs répétés comme ceux que nous avons à effectuer pour décrire le mouvement du trébuchet. De plus, nous sommes bien plus familiarisés avec ce langage de programmation que nous avons déjà beaucoup utilisé.
Cependant, nous avons finalement tranché en faveur de JavaScript, car ce langage, outre les nombreuses librairies de simulation qu'il propose, nous permet une animation du trébuchet de bien meilleure qualité et plus facile à maîtriser. En addition de cela, le JavaScript peut être utilisé avec du HTML (Hypertext Markup Language), ce qui nous offre l'opportunité de mettre la simulation dans une page web, ce qui serait impossible avec l'utilisation de Python.
Finalement, l'utilisation d'un nouveau langage nous permet de développer de plus larges compétences dans le domaine de la programmation.

Chapitre 1

Méthode

1.1 Théorie

La modélisation du trébuchet peut se diviser en deux grandes parties : dans un premier temps, il s'agit de simuler le mouvement du trébuchet en lui-même ; ensuite, il faut également parvenir à modéliser la trajectoire balistique du projectile.
Malgré la grande différence entre les notions physiques de ces deux problèmes, ils pourront, d'un point de vue mathématique, être exprimés de la même façon : à l'aide d'équations différentielles. Les solutions de ces équations seront les fonctions qui régissent le mouvement du projectile et du trébuchet dans le temps.
Pour déterminer les équations différentielles du mouvement du trébuchet, nous avons fait appel au formalisme lagrangien, tandis que pour celles de la trajectoire parabolique, nous avons utilisé les lois du mouvement de Newton.

1.1.1 Trajectoire balistique

Lorsque le projectile du trébuchet est libéré de la fronde, il poursuit une trajectoire balistique avec une vitesse initiale $\vec{v_i}$. Cette trajectoire se base sur la 2^{e} loi du mouvement de Newton, qui peut s'énoncer ainsi :

$$\sum \vec{F} = m\vec{a} \tag{1.1}$$

En principe, la trajectoire balistique d'un projectile se comporte comme un mouvement parabolique, c'est-à-dire que la particule se déplace à vitesse constante sur l'axe x (MRU) car aucune force ne s'applique sur l'horizontale, tandis que sur l'axe y, la masse est en proie à l'accélération gravitationnelle $\vec{g}$, ce qui l'entraîne vers le sol de manière uniformément accélérée (MRUA). En intégrant l'accélération selon le temps, on obtient un système d'équations pour la trajectoire du mouvement parabolique :

$$\begin{cases} x(t) = v_x t + x_i \\ y(t) = -\frac{1}{2} g t^2 + v_y t + y_i \end{cases}$$

Cependant, comme nous allons le constater, dans la réalité, des forces relatives à la mécanique des fluides s'appliquent sur le projectile, ce qui fait que l'accélération du boulet n'est en vérité pas uniforme au cours du temps. En effet, ces forces dépendent de la vitesse de la particule. Dans notre cas, il s'agit de la force de traînée ($\vec{F_t}$) et de l'effet Magnus ($\vec{F_M}$). Nous remarquerons aussi plus tard qu'elles influent également sur l'énergie du système car ce sont des forces dissipatives.

Voici un schéma qui permet d'illustrer un peu mieux nos propos :

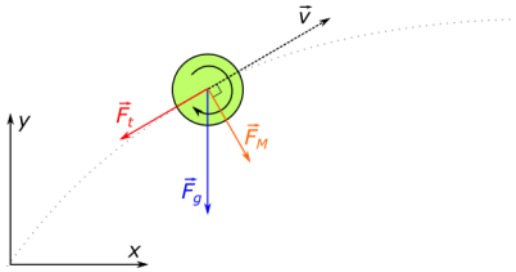


Figure 1: Forces qui s'appliquent sur le projectile [43]

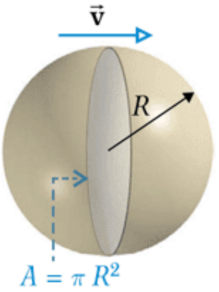
1.1.1.1 Force de traînée

Lorsqu'un corps est en mouvement dans un fluide, celui-ci subit une force de frottement colinéaire à la vitesse relative « $\vec{v}$ » de l'air par rapport au corps. Cette force, qu'on appelle *force de traînée* (F_t), est dépendante du module de la vitesse relative du fluide par rapport au corps v, de la masse volumique du fluide ρ, de la section efficace [1]du corps notée A, ainsi que d'un coefficient de frottement C_x.

$$F_t = \frac{1}{2} C_x A \rho v^2 \ [N] \tag{1.2}$$

Pour une balle sphérique de rayon r en déplacement dans un fluide de masse volumique ρ, C_x vaut 0.5 et A vaut $\pi r^2 [m^2]$.

$$F_{t(balle)}(r, \rho, v) = \frac{1}{2} \cdot 0.5 \cdot \pi r^2 \cdot \rho \cdot v^2 = \frac{1}{4} \cdot \pi r^2 v^2 \ [N]$$



(a) sphère

Figure 2: Schéma d'une sphère dans un fluide [22]

1.1.1.2 Effet Magnus

L'effet Magnus, découvert par le physicien allemand *Heinrich Gustave Magnus* (1802-1870), est un effet qui se produit lorsque qu'un corps sphérique est en rotation sur lui-même dans un fluide [16][43][26]. Cet effet peut être décrit comme une différence de pression entre deux points opposés de la sphère. Il a comme conséquence d'induire une force $\vec{F_M}$ perpendiculaire à la vitesse relative $\vec{v}$ du fluide. Cette force F_M est dépendante du rayon du corps r, de la vitesse v, de la vitesse

1. La **section efficace** d'un corps est la plus grande section qui coupe le corps perpendiculairement à la vitesse relative de l'air. Pour un corps sphérique, tel qu'une balle, la section efficace est une constante. Généralement, la section efficace change en fonction de l'orientation du corps dans l'air qui l'entoure.

angulaire de rotation ω exprimée en $\left[\frac{rad}{s}\right]$ et de la masse volumique du fluide ρ. Dans le cadre de ce travail, nous considérons que le vecteur de rotation $\vec{\omega}$ [2] est perpendiculaire au vecteur vitesse $\vec{v}$.

$$F_M = \frac{1}{2}\pi r^3 \rho \omega v \ [N] \tag{1.3}$$

1.1.2 Mouvement du trébuchet

1.1.2.1 Méthode de Lagrange

L'équation de Lagrange est une reformulation des équations du mouvement de Newton. Cette équation permet d'exprimer la dynamique du trébuchet à l'aide des bilans d'énergies potentielle et cinétique du système. En outre, l'utilisation du lagrangien permet d'éviter une approche vectorielle, particulièrement lourde en notation, que nous imposerait l'utilisation des moments de forces [34]. L'équation se présente ainsi :

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} = 0 \tag{1.4}$$

Où L, le lagrangien, correspond à la différence entre l'énergie cinétique, notée T, et l'énergie potentielle du système, notée V.

$$L = T - V$$

Coordonnées généralisées : l'utilisation des coordonnées cartésiennes dans un système complexe comme le trébuchet se révèle inefficiente puisque les positions changent constamment et les projections sur un référentiel orthonormé seraient bien trop imposantes. Les coordonnées généralisées peuvent représenter les contraintes du système sans utiliser la position. Les N coordonnées correspondent aux N degrés de liberté du système [12].

$$q_i = (q_1, \ldots, q_n)$$

Dans notre cas, il s'agit des angles formés par les bras de l'engin, à savoir ; α, β, et γ, ce qui nous donne les trois équations de Lagrange pour notre système :

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\alpha}}\right) - \frac{\partial L}{\partial \alpha} = 0 \tag{1.5}$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\beta}}\right) - \frac{\partial L}{\partial \beta} = 0 \tag{1.6}$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\gamma}}\right) - \frac{\partial L}{\partial \gamma} = 0 \tag{1.7}$$

En effet, le système que nous souhaitons simuler peut se représenter uniquement avec trois angles qui varient au cours du temps ; le mouvement des bras et des masses se déduisent aisément par la suite.

1.1.3 Méthodes d'intégration numérique

Comme décrit précédemment, la description mathématique de notre modèle repose sur des équations différentielles. Or, la solution analytique d'une équation différentielle, c'est-à-dire, une fonction qui correspond *parfaitement* à notre équation, ne peut parfois pas être déterminée. Dans la situation où aucune fonction répondant aux contraintes de notre système n'a été trouvée, nous

2. Le **vecteur** $\vec{\omega}$ est un vecteur perpendiculaire au plan de rotation. Nous considérons que sa norme est positive lorsque la balle est en rotation vers l'avant et donc négative lorsque la balle est en rotation vers l'arrière (en fonction de l'axe x).

devons estimer la solution à l'aide d'une intégration numérique.
L'intégration numérique va approximer la solution avec les informations qu'elle a à sa disposition, les dérivées de la fonction. À l'aide de ces dernières, on peut estimer une intégration, ce qui nous permet d'approximer la fonction recherchée.
Dès lors, il existe plusieurs méthodes pour effectuer une intégration numérique. Certaines étant plus précises que d'autres, nous avons décidé de tester trois méthodes d'intégration différentes afin d'observer quel impact elles auraient sur la simulation. Nous avons donc choisi les méthodes d'Euler, de Heun, ainsi que de Runge-Kutta (4^e ordre) [4]. Afin de mieux comprendre, nous considérons une fonction dépendante de la variable x dont nous connaissons la fonction de la pente $f(x)$ en un certain point $(x_0\ ; y_0)$, ainsi qu'un pas h qui définit directement la position du point suivant $(x_0 + h\ ; y_1)$.

1.1.3.1 Méthode d'Euler

La méthode d'Euler est la méthode que nous avons initialement utilisée, elle est aussi la plus simple à mettre en place. Elle est équivalente à la méthode de Runge-Kutta d'ordre 1. Elle consiste à considérer que la dérivée de la fonction est constante lorsque h est assez petit. Cela nous permet de déduire la position approximée en y du prochain point de la fonction [6][15].

$$\begin{aligned} x_{n+1} &= x_n + h \\ p_n &= f(x_n, y_n) \\ y_{n+1} &= y_n + p_n h \end{aligned}$$

Cette méthode est d'ordre 1 car l'erreur totale accumulée est d'ordre h.

1.1.3.2 Méthode de Heun

La méthode de Heun est un cas précis de la méthode de Runge-Kutta d'ordre 2. Celle-ci consiste à faire une moyenne entre la dérivée du point actuel et la dérivée du point suivant, calculée avec la méthode d'Euler [19].

$$\begin{aligned} x_{n+1} &= x_n + h \\ p_n &= f(x_n, y_n) \\ q_n &= f(x_n + h, y_n + p_n h) \\ y_{n+1} &= y_n + \frac{p_n + q_n}{2} \cdot h \end{aligned}$$

Cette méthode est d'ordre 2 car l'erreur totale accumulée est d'ordre h^2.

1.1.3.3 Méthode de Runge-Kutta (4^e ordre)

La méthode de Runge-Kutta d'ordre 4 utilise des outils de lissage encore plus précis [25]. En étant concis, elle consiste à prendre la dérivée de notre point de départ p_n puis avec une méthode d'Euler de pas $\frac{h}{2}$ nous obtenons la dérivée q_n d'un point milieu de l'intégration. Ensuite, de nouveau depuis notre point de départ, nous réapproximons avec une méthode d'Euler, en utilisant la dérivée q_n et un pas de $\frac{h}{2}$, un nouveau point milieu dont la dérivée sera r_n. Finalement nous refaisons une approximation d'Euler avec r_n et un pas h pour trouver la dernière dérivée s_n. Grâce

à ces différents points nous pourrons estimer précisément le point suivant de la fonction [30].

$$\begin{aligned} x_{n+1} &= x_n + h \\ p_n &= f(x_n, y_n) \\ q_n &= f(x_n + \frac{h}{2}, y_n + p_n \frac{h}{2}) \\ r_n &= f(x_n + \frac{h}{2}, y_n + q_n \frac{h}{2}) \\ s_n &= f(x_n + h, y_n + r_n h) \\ y_{n+1} &= y_n + \frac{p_n + 2q_n + 2r_n + s_n}{6} \cdot h \end{aligned}$$

Cette méthode est d'ordre 4 car l'erreur totale accumulée est d'ordre h^4.

1.2 Schéma et données

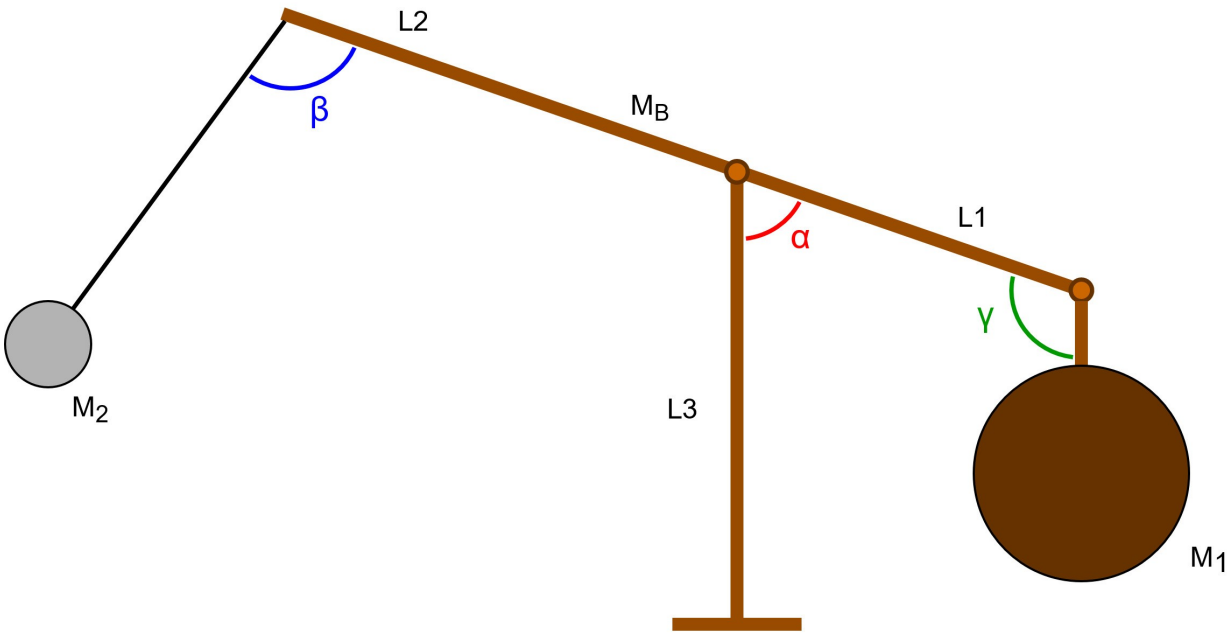


Figure 3: Données du trébuchet

Coordonnées généralisées		
α	alpha	[rad]
β	beta	[rad]
γ	gamma	[rad]
Constantes du trébuchet		
m_1	masse du contrepoids	[kg]
m_2	masse du projectile	[kg]
m_b	masse du balancier	[kg]
l_1	longueur du bras court	[m]
l_2	longueur du bras long	[m]
l_3	hauteur du pivot	[m]
l_4	longueur du contrepoids	[m]
l_5	longueur de la fronde	[m]
l_b	longueur du centre de masse du balancier au point de pivot	[m]
Constantes de la balistique		
$releaseAngle$	angle de libération	[°]
$vvent$	vitesse du vent	[m/s]
$ivent$	inclinaison du vent	[°]
ω	vitesse de rotation du projectile	[°/s]

Après mûre réflexion, nous avons choisi de considérer la masse du balancier, ceci pour stabiliser le modèle, dans le cas où nous voudrions continuer la simulation du trébuchet après que le projectile soit parti. De plus, la rajouter dans les équations de Lagrange n'est pas si compliqué car m_b ne dépend que de α pour son mouvement. Il est à noter que nous négligeons la torsion de la structure et nous considérons que la masse du balancier est située au milieu du bras principal, puisque nous estimons que la verge a une densité homogène : $l_b = \frac{l_1+l_2}{2}$. De plus, nous considérons que la fronde se comporte comme un bras de bois incompressible.

1.3 Calculs

Afin de rester concis, les calculs des équations de Lagrange seront mis en annexe du document.

1.3.1 Calcul de l'accélération du projectile en chute libre

Pour calculer l'accélération de la balle en chute libre, nous utilisons la 2e loi de Newton. Dans les calculs ci-dessous, θ correspond à l'inclinaison relative de l'air par rapport à l'horizontale (axe x), et m est la masse du projectile.
Voici vectoriellement la somme des forces qui s'appliquent sur le projectile :

$$\sum \vec{F} = \vec{F}_t + \vec{F}_M + \vec{F}_g = m \cdot \vec{a}$$

Nous commençons à calculer la vitesse relative de l'air par rapport à la balle. Pour cela, nous aurons affaire à deux variables v_x, v_y qui correspondent à la vitesse de la balle en x et en y ainsi que deux constantes : la vitesse du vent v_{vent} et l'inclinaison du vent i_{vent}.

$$\vec{v}_{relative}(v_x, v_y) = \begin{pmatrix} -v_x + v_{vent} \cdot \cos(i_{vent}) \\ -v_y + v_{vent} \cdot \sin(i_{vent}) \end{pmatrix}$$

Grâce à ces deux valeurs, nous pouvons déduire l'inclinaison de la vitesse relative θ.

$$\theta = \begin{cases} tan^{-1}\left(\frac{v_{relative;y}}{v_{relative;x}}\right), & \text{si } v_{relative;x} \geq 0 \\ tan^{-1}\left(\frac{v_{relative;y}}{v_{relative;x}}\right) + \pi, & \text{si } v_{relative;x} < 0 \end{cases}$$

Nous savons que la force de traînée ($\vec{F_t}$) ainsi que de la force due à l'effet Magnus ($\vec{F_M}$) ne dépendent que d'une seule variable qui est la vitesse relative de l'air par rapport au corps, elle-même dépendante de la vitesse de la balle. Il nous suffit donc d'utiliser les fonctions (1.2) et (1.3) afin de résoudre le système.
Nous commençons par calculer la somme des forces qui s'appliquent en x avec laquelle nous déduisons l'accélération en x :

$$\sum F_x = F_t \cdot \cos(\theta) - F_M \cdot \sin(\theta) = m \cdot a_x$$

$$\Rightarrow a_x = \frac{\sum F_x}{m} = \frac{F_t \cdot \cos(\theta) - F_M \cdot \sin(\theta)}{m}$$

$$= \frac{\frac{1}{4}\pi r^2 \rho v_{relative}^2 \cdot \cos(\theta) - \frac{1}{2}\pi r^3 \rho \omega v_{relative}^2 \cdot \sin(\theta)}{m}$$

Nous faisons de même pour y :

$$\sum F_y = F_t \cdot \sin(\theta) + F_M \cdot \cos(\theta) - F_g = m \cdot a_y$$

$$\Rightarrow a_y = \frac{\sum F_y}{m} = \frac{F_t \cdot \sin(\theta) + F_M \cdot \cos(\theta)}{m} - g$$

$$= \frac{\frac{1}{4}\pi r^2 \rho v_{relative}^2 \cdot \sin(\theta) + \frac{1}{2}\pi r^3 \rho \omega v_{relative}^2 \cdot \cos(\theta)}{m} - g$$

La vitesse relative de l'air dépendant de la vitesse de la balle, nous pouvons remarquer que le calcul de l'accélération dépend uniquement de deux variables : les vitesses v_x et v_y. Les constantes telles que v_{vent}, i_{vent} ou encore ω seront définies par les utilisateurs de notre programme. Pour les autres constantes ρ et r, nous les définirons nous-mêmes dans le code.
Nous pouvons donc déduire que l'accélération à un certain pas ne dépend que de la vitesse de la balle. Les 2 fonctions de l'accélération peuvent donc s'écrire ainsi : $a_x(v_x, v_y)$ et $a_y(v_x, v_y)$. Nous les réduirons en $a(v_x, v_y)$, une fonction qui retourne le vecteur accélération $\begin{pmatrix} a_x \\ a_y \end{pmatrix}$.
Cette fonction retourne le résultat sous forme de dictionnaire dans le code afin de facilement sélectionner l'accélération en x ou en y. Dans la suite de nos calculs, nous attribuerons l'indice x ou y à la fonction pour en définir soit a_x, soit a_y.

1.3.2 Calcul du lagrangien et des équations différentielles

Nous commençons par calculer le lagrangien L du système :

$L = \frac{1}{2}[m_1({l_1}^2 + {l_4}^2) + m_b {l_b}^2 - 2m_1 l_1 l_4 \cos(\gamma) + m_2({l_2}^2 + {l_5}^2) - 2m_2 l_2 l_5 \cos(\beta)]\dot{\alpha}^2 + \frac{1}{2}m_1 {l_4}^2 \dot{\gamma}^2 + \frac{1}{2}m_2 {l_5}^2 \dot{\beta}^2 + m_1({l_4}^2 - l_1 l_4 \cos(\gamma))\dot{\alpha}\dot{\gamma} + m_2(-{l_5}^2 + l_2 l_5 \cos(\beta))\dot{\alpha}\dot{\beta} - g(-m_1 l_1 + m_2 l_2 + m_b l_b)\cos(\alpha) + g m_2 l_5 \cos(\beta - \alpha) - g m_1 l_4 \cos(\gamma + \alpha)$

Nous développons les équations de Lagrange (1.4) avec les 3 coordonnées généralisées du système : α, β, γ. Cela nous donne un système de trois équations différentielles ordinaires, ou EDO [3],

3. Equation différentielle où la solution, si on parvient à la déterminer, est une fonction qui a une relation spécifique à ses dérivées. Contrairement aux équations différentielles partielle (EDP), les EDO ne dépendent que d'une seule variable, dans notre cas, le temps.

de second ordre[4]. Ces équations nous permettront ensuite de déterminer les variations angulaires dans le temps, et donc, le mouvement du trébuchet.

1) $\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\alpha}}\right) - \frac{\partial L}{\partial \alpha} = 0$

$\left[m_1({l_1}^2 + {l_4}^2) + m_b{l_b}^2 - 2m_1l_1l_4\cos(\gamma) + m_2({l_2}^2 + {l_5}^2) - 2m_2l_2l_5\cos(\beta)\right]\ddot{\alpha} + m_2(-{l_5}^2 + l_2l_5\cos(\beta))\ddot{\beta} + m_1({l_4}^2 - l_1l_4\cos(\gamma))\ddot{\gamma} + (2m_2l_2l_5\dot{\beta}\sin(\beta) + 2m_1l_1l_4\dot{\gamma}\sin(\gamma))\dot{\alpha} + m_1l_1l_4\sin(\gamma)\dot{\gamma}^2 - m_2l_2l_5\sin(\beta)\dot{\beta}^2 - g(-m_1l_1 + m_2l_2 + m_bl_b)\sin(\alpha) - gm_2l_5\sin(\beta - \alpha) - gm_1l_4\sin(\gamma + \alpha) = 0$

2) $\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\beta}}\right) - \frac{\partial L}{\partial \beta} = 0$

$m_2(-{l_5}^2 + l_2l_5\cos(\beta))\ddot{\alpha} + m_2{l_5}^2\ddot{\beta} - m_2l_2l_5\sin(\beta)\dot{\alpha}^2 + gm_2l_5\sin(\beta - \alpha) = 0$

3) $\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\gamma}}\right) - \frac{\partial L}{\partial \gamma} = 0$

$m_1({l_4}^2 - l_1l_4\cos(\gamma))\ddot{\alpha} + m_1{l_4}^2\ddot{\gamma} - m_1l_1l_4\sin(\gamma)\dot{\alpha}^2 - gm_1l_4\sin(\gamma + \alpha) = 0$

Ces équations sont implémentées de cette manière :

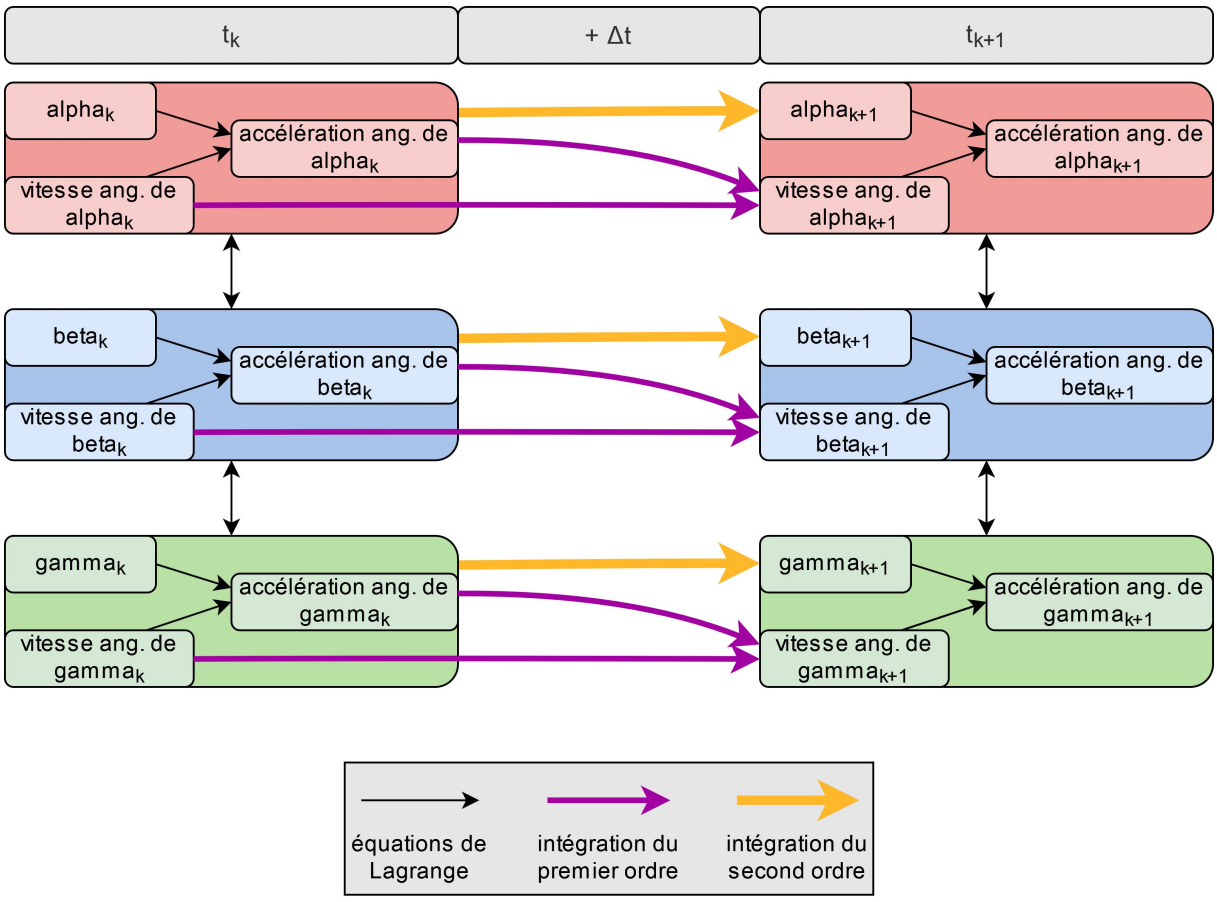


Figure 4: Mécanisme d'intégration des équations du trébuchet

Il est à noter que les valeurs obtenues au cours du temps dépendent fortement des conditions initiales du système, car c'est à partir d'elles que la suite est calculée.
Puisque cette intégration est une approximation, à chaque étape, l'estimation s'écarte petit à

4. Cela signifie que le degré maximale de dérivation de la fonction est de deux, dans notre modèle, il s'agit des accélérations angulaires $\ddot{\alpha}, \ddot{\beta}$ et $\ddot{\gamma}$.

petit de la solution réelle, on dit que la suite *diverge*. Après un certain temps t, la modélisation n'est plus valable.
Pour la simulation de notre trébuchet, les accélérations angulaires à un instant t nous permettent d'approximer les angles et les vitesses angulaires à un temps $t+\Delta t$, c'est pourquoi il est pratique de réécrire les équations sous une forme linéaire ayant pour inconnues $\ddot{\alpha}$, $\ddot{\beta}$ et $\ddot{\gamma}$. Les trois équations correspondent aux trois EDO.

$$c_{11}\ddot{\alpha} + c_{12}\ddot{\beta} + c_{13}\ddot{\gamma} = w_1 \tag{1.8}$$

$$c_{21}\ddot{\alpha} + c_{22}\ddot{\beta} + c_{23}\ddot{\gamma} = w_2 \tag{1.9}$$

$$c_{31}\ddot{\alpha} + c_{32}\ddot{\beta} + c_{33}\ddot{\gamma} = w_3 \tag{1.10}$$

Où les coefficients sont :

$c_{11} = m_1(l_1^2 + l_4^2) + m_b l_b^2 - 2m_1 l_1 l_4 \cos(\gamma) + m_2(l_2^2 + l_5^2) - 2m_2 l_2 l_5 \cos(\beta)$
$c_{12} = m_2(-l_5^2 + l_2 l_5 \cos(\beta))$
$c_{13} = m_1(l_4^2 - l_1 l_4 \cos(\gamma))$
$w_1 = (-m_1 l_1 + m_2 l_2 + m_b l_b) g \sin(\alpha) + m_2 l_5 g \sin(\beta - \alpha) + m_1 l_4 g \sin(\gamma + \alpha) - 2(m_1 l_1 l_4 \sin(\gamma)\dot{\gamma} + m_2 l_2 l_5 \sin(\beta)\dot{\beta})\dot{\alpha} - m_1 l_1 l_4 \sin(\gamma)\dot{\gamma}^2 + m_2 l_2 l_5 \sin(\beta)\dot{\beta}^2$

$c_{21} = m_2(-{l_5}^2 + l_2 l_5 \cos(\beta)) = c_{12}$
$c_{22} = m_2 {l_5}^2$
$c_{23} = 0$
$w_2 = m_2 l_2 l_5 \sin(\beta)\dot{\alpha}^2 - m_2 l_5 g \sin(\beta - \alpha)$

$c_{31} = m_1({l_4}^2 - l_1 l_4 \cos(\gamma)) = c_{13}$
$c_{32} = 0$
$c_{33} = m_1 {l_4}^2$
$w_3 = m_1 l_1 l_4 \sin(\gamma)\dot{\alpha}^2 + m_1 l_4 g \sin(\gamma + \alpha)$

Cette forme nous permet de résoudre le système d'équations à l'aide d'une inversion de matrice que nous ferons à chaque étape.

$$\begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{12} & c_{22} & 0 \\ c_{13} & 0 & c_{33} \end{bmatrix} \cdot \begin{bmatrix} \ddot{\alpha} \\ \ddot{\beta} \\ \ddot{\gamma} \end{bmatrix} = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} \Rightarrow \begin{bmatrix} \ddot{\alpha} \\ \ddot{\beta} \\ \ddot{\gamma} \end{bmatrix} = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} \cdot \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{12} & c_{22} & 0 \\ c_{13} & 0 & c_{33} \end{bmatrix}^{-1}$$

Pour inverser cette matrice, nous utilisons la méthode d'élimination de Gauss-Jordan.
Nous pouvons dès lors définir la fonction $calculsAccAngu(\alpha, \beta, \gamma, \dot{\alpha}, \dot{\beta}, \dot{\gamma})$ qui retourne les accélérations angulaires sous forme d'une matrice.

1.3.2.1 Élimination de Gauss-Jordan

Sans entrer trop dans les détails du fonctionnement de ce pivot, cette méthode permet, à l'aide d'échanges, de multiplications et d'additions de lignes, d'inverser une matrice [42][17][31]. Étant donné que la programmation de ce pivot ne présente que peu d'intérêt dans le cadre de notre travail, nous avons fait appel à un code source provenant d'Internet [1]. Cette partie du programme est donc la seule n'ayant pas été rédigée par nous.

1.3.3 Collision de la fronde ou du contrepoids avec le bras du trébuchet

Dans notre travail, le système du trébuchet n'est pas réellement décrit correctement dans son ensemble. En effet, si nous nous intéressons au mouvement du trébuchet sur l'entièreté de la simulation, nous nous devons donc de prendre en compte les collisions entre les différents bras. Celles-ci apparaissent généralement après que le projectile a été libéré du fait que la fronde se retrouve avec une masse quasi nulle. Cela amène l'intégration numérique à diverger bien plus rapidement puisque le moment de force conduit à des accélérations angulaires immenses.
Nous faisons donc une grossière estimation de ces collisions car ils nous faudraient entièrement résoudre à nouveau le système en utilisant des méthodes plus complexes et plus longues à mettre en place si nous voulions être très précis. Ce qui, nous devons l'avouer, nécessiterait sûrement un deuxième travail de Maturité.

La manière dont nous opérons n'est pas très complexe : nous vérifions si la prochaine itération de l'angle β ou γ, c'est-à-dire β_{n+1} ou γ_{n+1}, sort de l'intervalle $]0; 2\pi[$. Si c'est le cas, il y a bien collision. à la prochaine itération, nous changeons donc le signe la vitesse angulaire qui sort de l'intervalle , c'est-à-dire $\dot{\beta}_n$ ou $\dot{\gamma}_n$. Nous multiplions ensuite cette vitesse par un certain coefficient de restitution que nous nommerons e compris entre 0 et 1. Dans le cas où ce coefficient est inférieur à 1, cela a pour effet d'engendrer une perte d'énergie au système. Dans notre programme, nous avons décidé de prendre le coefficient de restitution e du bois puisque le bras principal du trébuchet est en bois. e vaut donc 0,5.
Ces deux formules peuvent résumer notre méthode :

$$\beta_{n+1} \notin]0; 2\pi[\Rightarrow \dot{\beta}'_n = -e \cdot \dot{\beta}_n \ , \quad e \in [0; 1]$$

$$\gamma_{n+1} \notin]0; 2\pi[\Rightarrow \dot{\gamma}'_n = -e \cdot \dot{\gamma}_n \ , \quad e \in [0; 1]$$

Cela aura pour effet de faire augmenter ou diminuer les angle β et γ afin de les empêcher de sortir de l'intervalle $]0; 2\pi[$.
L'intérêt principal de cet outil, en plus d'ajouter de la véracité à notre simulation, étant qu'il permet de stabiliser le modèle lorsque les angles deviennent trop grands. En effet, l'intégration numérique, surtout la méthode d'Euler, a tendance à extrapoler les mouvements et, en conséquence, à augmenter l'énergie du système. En provoquant ces pertes contôlées d'énergie, notre système se maintient stable après le lancé de m_2
Nous pouvons dès lors nous questionner sur la perte d'énergie induite au système car cela peut aussi nous renseigner sur la qualité de l'approximation des collisions. Ces calculs d'énergies sont mis en annexe.

Chapitre 2

Programmation

Dans cette partie, nous allons expliquer la structure générale du code, expliquer les raisons de certains choix ainsi que décrire le fonctionnement de fonctionnalités supplémentaires que nous avons implémentées. L'intégralité du code est disponible au téléchargement ici :

https://github.com/AdrienB2/TM_trebuchet

2.1 Outils

Comme expliqué dans l'introduction, nous avons choisi d'utiliser du JavaScript pour réaliser ce projet. JavaScript est un langage de programmation qui s'exécute, dans la majorité des cas, sur des pages internet côté client et doit donc obligatoirement être lié à un fichier HTML pour être exécuté dans un navigateur internet. Afin d'améliorer la qualité esthétique de l'affichage, nous avons également utilisé CSS (Cascading Style Sheets) pour la mise en forme.
En réalisant ce projet, nous n'avons eu besoin que d'une seule bibliothèque de code nommée *chart.js*[1] pour l'affichage des graphiques, ce qui nous a grandement facilité le travail. Tout le reste est effectué à l'aide des fonctions de base des trois langages cités, car cela est largement suffisant pour ce que nous voulions faire. Même si l'utilisation de certaines bibliothèques de code nous aurait peut-être aidé pour certaines parties, le choix d'utiliser un minimum de bibliothèques tierces a été fait pour des raisons d'optimisation du code. En effet, charger une bibliothèque de code entière juste pour une ou deux fonctions que nous pouvons coder nous-mêmes facilement augmente le temps de chargement du site et demande plus de mémoire vive, ce qui peut créer de forts ralentissements sur des ordinateurs en ayant peu.

2.2 Structure du code

Voici la structure globale de notre code source :

```
****************************** LISTE DES FONCTIONS ******************************

0. PETITES ET MOYENNES FONCTIONS :
        0.1. "miax(liste)"
                Calcul du maximum et du minimum d'une liste.
```

1. https://www.chartjs.org/

```
        0.2. "inc(vx,vy)"
                Fonction qui donne l'inclinaison du projectile.
        0.3. "matrix_invert(M)"
                Inversion de matrice
        0.4. "calculsAccAngu(alpha,beta,gamma,d1alpha,d1beta,d1gamma)"
                Calcul des accélérations angulaires.

1. FONCTIONS POUR LES CALCULS PRINCIPAUX (utilisées dans la fonction "calculate")
        1.1. Méthodes d'intégration numérique pour les angles
                1.1.1. "EulerMethod(dt, PrésenceBalle)"
                        Méthode d'Euler.
                1.1.2. "Heun(dt, PrésenceBalle)"
                        Méthode d'Heun.
                1.1.3. "RungeKutta(dt, PrésenceBalle)"
                        Méthode de Runge-Kutta (RK4).
        1.2. "balistique(vi,incl,vvent,ivent,xi,   yi)"
                Calculs de la trajectoire du projectile.

2. FONCTION POUR L'AFFICHAGE (utilisée dans la fonction "Display")
        2.1. "draw(alpha, beta, gamma, pos, vitesse,i)"
                Affiche le trébuchet et la trajectoire de la balle.

3. FONCTIONS PRINCIPALES
        3.1. "calculate()"
                Calcul de la simulation.
        3.2. "Scale()"
                Fonction qui détermine le bon coefficient d'affichage.
        3.3. "displaySim(i,slowMotion)"
                Affiche le résultat de la simulation sur le canvas.
        3.4. "Initialisation()"
                Fonction qui vérifie et modifie les données
                avant le lancement de la simulation

4. FONCTIONS FINALES RELIÉES CHACUNE A UN BOUTON
        4.1. "startSim()"
                Lance la simulation.
        4.2. "Stop()"
                Stoppe définitivement la simulation
        4.3. "OnOff()"
                Arrête ou relance la simulation.
        4.4. Ralenti
                Fonctions qui, ensemble, permettent de ralentir la simulation.
                4.4.1. "slowMotion()"
                        Bouton "Ralenti/Normal".
                4.4.2. "RalentiCursorFunction()"
                        Curseur de 0 à 10 pour le coefficient de ralenti "ral".
        4.5. "Resize()"
                Fonction qui redimensionne la simulation.

****************** HIÉRARCHIE DES FONCTIONS (dans "startSim") ******************
```

```
4.1. "startSim()"
        3.4. "Initialisation()"
        3.1. "calculate()"
                1.1.3. "RungeKutta(dt, PrésenceBalle)"
                        0.4. "calculsAccAngu(alpha,beta,gamma,d1alpha,d1beta,d1gamma)"
                                0.3. "matrix_invert(M)"
                1.2. "balistique(vi, incl, vvent, ivent, xi, yi)"
                        0.2. "inc(vx,vy)"
        3.2. "Scale()"
                0.1. "miax(liste)"
        3.3. "displaySim(i,slowMotion)"
                2.1. "draw(alpha, beta, gamma, pos, vitesse,i)"
```

2.3 Simulation

Le code peut être divisé en deux grandes parties :

1. Le calcul des angles du trébuchet et de la position du projectile pour toute la durée de la simulation.
2. L'affichage de la simulation animée à l'écran.

Ce choix a été fait pour assurer la fluidité de l'affichage. En effet, après avoir effectué des tests, nous avons remarqué que si les calculs étaient effectués en même temps que l'affichage, ce dernier était saccadé car le processeur n'avait pas le temps de faire tous les calculs entre deux images, de plus, cela pouvait faire boguer le navigateur et ralentir l'intégralité de l'ordinateur de l'utilisateur.
Une fois les calculs finis et les résultats enregistrés dans des listes, il suffit au programme de lire les données les unes après les autres et de les afficher à l'utilisateur. Cela ne demande pas énormément de puissance au processeur et permet l'utilisation de la simulation même sur des ordinateurs ayant une faible puissance de calcul.

2.3.1 Calculs

Comme expliqué plus tôt, la modélisation du trébuchet peut se diviser en deux parties, le mouvement du trébuchet et la trajectoire du projectile. La partie calculs du code se divise de la même façon. Cependant, elle est composée de six fonctions principales : deux pour chacune des méthodes de calcul : Euler (RK1), Heun (RK2) et Runge-Kutta (RK4).
Nous avons laissé la possibilité à l'utilisateur de choisir l'une d'elle dans l'interface. Ces fonctions permettent de calculer les angles du trébuchet pour le pas de temps suivant en prenant les angles du pas de temps actuel. Nous appelons donc la fonction choisie dans une boucle jusqu'à ce que l'angle de libération choisi soit atteint. À chaque fois que la fonction sélectionnée est appelée, nous la ré-appelons avec les résultats que cette dernière nous a donnés pour le pas de temps précédant. En répétant ce processus jusqu'à l'angle de libération, nous obtenons toutes les valeurs pour tous les pas de temps.
Une fois le calcul du mouvement du trébuchet terminé, nous appelons une autre fonction dans une boucle pour calculer la partie balistique. Elle permet de calculer la position et la vitesse du projectile. Cette fonction est appelée de la même manière que la précédente, dans une boucle en changeant les paramètres chaque fois que la fonction est appelée et ce jusqu'à ce que le projectile atteigne le sol.
Une fois la balistique calculée, nous en avons presque terminé avec les calculs. En effet, nous souhaitons encore continuer de simuler le mouvement du trébuchet une fois le projectile lancé. Cela n'est pas très compliqué car nous avons déjà les fonctions et que nous connaissons le nombre de pas de temps que contient notre simulation. Nous pouvons donc appeler la fonction le même

nombre de fois que la fonction qui calcule la balistique. Ainsi, nous avons deux listes de même longueur qui pourront être lue jusqu'à la fin de la simulation.

2.3.2 Affichage de la simulation

La simulation est affichée sur un élément HTML nommé `canvas`. Cet élément permet de dessiner des formes sur une page internet en JavaScript. Nous avons créé une fonction qui prend en paramètre les angles du trébuchet, la position du projectile, les données à afficher au-dessus de la simulation, et un coefficient d'agrandissement pour dessiner sur le `canvas` le trébuchet, le projectile, la trajectoire des projectiles précédents et afficher en temps réel les données. Cette fonction est appelée à l'aide de la méthode `setInterval()` qui permet d'exécuter du code à un intervalle de temps précis. Cette méthode continue d'exécuter le code jusqu'à ce que `clearInterval()` soit appelé. Cela nous permet donc d'afficher chaque image de la simulation l'une après l'autre tout en choisissant l'intervalle de temps, ce qui nous offre la possibilité d'afficher la simulation à vitesse réelle, en accéléré ou en ralenti.
Une fois que la simulation est terminée, la méthode `clearInterval()` stoppe l'affichage de la simulation sans pour autant effacer tous les éléments du `canvas`.

2.3.3 Lien entre l'interface et le code

Maintenant que nous avons tout cela, il nous faut un moyen pour permettre à un utilisateur de rentrer des valeurs et de lancer la simulation facilement. Pour cela, nous avons créé un formulaire avec tous les paramètres que l'on peut modifier ainsi qu'un bouton pour lancer la simulation. Lorsque l'on clique sur ce dernier, une fonction est appelée dans le code. Cette fonction récupère les données entrées dans le formulaire et vérifie que les valeurs soient valides pour que la simulation puisse fonctionner correctement. Ensuite, la fonction appelle la fonction de la simulation du trébuchet dans une boucle jusqu'à ce que l'angle de libération soit atteint. Puis la fonction de calcul de la balistique jusqu'à ce que le projectile atteigne le sol. Ensuite la fonction qui calcule le mouvement du trébuchet est appelée le nombre de fois qu'il faut pour que la simulation du trébuchet et la simulation de la balistique soient de la même longueur. Une fois tous les calculs terminés, la méthode `setInterval()` appelle la fonction qui affiche une image de la simulation.

2.4 Fonctionnalités supplémentaires

A ce stade, le site fonctionne parfaitement, mais l'interface est très difficile à utiliser. Nous avons donc travaillé dessus et sur l'ajout de fonctionnalités supplémentaires.

2.4.1 Affichage de la simulation au ralenti

Changer la vitesse de lecture de la simulation a été assez simple à mettre en place. En effet, l'animation de la simulation étant basée sur les calculs de la simulation, il nous suffit de changer la vitesse à laquelle nous rafraîchissons l'affichage : pour ralentir la simulation, nous affichons moins d'images par seconde et pour l'accélérer, nous en affichons plus. Cette approche n'est toutefois pas parfaite car le nombre d'images par seconde change, ce qui influence également la fluidité de l'affichage.

2.4.2 Changement de focalisation

Une autre fonction que nous avons décidé d'implémenter permet à l'utilisateur de choisir deux différents modes de focalisation sur la simulation ; soit un gros plan sur le trébuchet, soit une vue d'ensemble de la trajectoire. Cela nous permet à la fois d'observer la trajectoire balistique

et le mouvement du trébuchet après le lancer.
Nous avons pu implémenter cette fonctionnalité de façon très simple car notre fonction d'affichage de la simulation prend en compte un paramètre de zoom (`scale`) pour modifier la taille d'affichage de la simulation. Nous avons donc juste codé une fonction qui modifie le coefficient de zoom pour cadrer l'affichage sur le trébuchet. Par défaut, quand on lance la simulation, le `scale` permet d'avoir une vue rapprochée du trébuchet et, une fois le projectile parti du trébuchet, le `scale` diminue jusqu'à ce que l'intégralité de la trajectoire du projectile soit visible. Nous avons également ajouté un bouton permettant de rester en gros plan sur le trébuchet.

2.4.3 Graphiques

Nous nous sommes dit que cela serait pertinent et utile d'ajouter un moyen de voir l'évolution des angles, de la position, de la vitesse du projectile et de l'énergie tout au long de la simulation. Nous avons donc ajouté un onglet sur lequel toutes ces informations sont disponibles sous forme de graphiques. Pour afficher ces graphiques, nous avons utilisé une bibliothèque de code appelée `chart.js`. Celle-ci nous permet d'afficher de façon très simple les graphiques des différentes valeurs au cours du temps avec du code JavaScript.

2.4.4 Adaptation aux différentes tailles d'écran

En faisant des tests nous nous sommes retrouvés face à un problème : si le site fonctionnait très bien sur des grandes tailles d'écran, comme sur un ordinateur par exemple, cela n'était pas du tout le cas sur des écrans de petite taille. Nous avons donc dû ajuster la mise en forme du site en fonction de la taille de l'écran. Pour faire cela, nous avons utilisé la règle `@media` qui permet d'appliquer un style différent en fonction du type d'appareil sur lequel le site est utilisé. Dans notre cas, nous avons utilisé `@media only screen and (max-width: 1250px)`. Cette ligne dans un fichier CSS permet d'ordonner au navigateur d'appliquer le style uniquement si le site est affiché sur un écran de moins de 1250 pixel de large. En en définissant plusieurs pour les différentes tailles d'écran, nous pouvons rendre le site compatible avec le plus d'appareils différents possible. Comme cette méthode est longue et peut vite se révéler limitée pour certains changements importants sur de très petites tailles d'écrans, nous avons décidé que cela n'avait pas de réelle utilité d'adapter le site pour des écrans plus petits qu'une petite tablette.

2.4.5 Affichage des trajectoires précédentes

Nous nous sommes dit qu'il serait intéressant de pouvoir comparer les trajectoires entre les différents lancers. Pour cela, toutes les trajectoires sont enregistrées dans une liste du nom d'`animPoints` ce qui nous permet de les ré-afficher pendant le lancer suivant. Les trajectoires précédentes sont affichées dans une couleur plus claire pour pouvoir les différencier de la dernière trajectoire et rendre l'interface plus esthétique.

2.4.6 Système d'onglets

Quand nous avons décidé d'ajouter les graphiques sur le site, nous nous sommes demandé où les afficher. Notre choix a été de les mettre dans un autre onglet. Nous avions donc besoin de coder un système qui nous permette de changer tout ce qui est affiché sur le site en cliquant sur un lien. Pour ce faire, le contenu d'un onglet a été placé intégralement à l'intérieur d'une `<div> <\div>`, un élément HTML qui permet de grouper plusieurs éléments, ce qui nous permettra d'afficher le contenu de l'onglet que nous voulons voir et de masquer les autres. Nous avons donné à chacune de ces `<div>` un `id` différent pour pouvoir facilement les afficher et les masquer avec JavaScript. Nous avons aussi ajouté l'attribut `data-tab-content` [2] afin de pouvoir facilement

2. Les attributs HTML commençant par `data-` sont des attributs personnalisés qui permettent de passer des paramètres que l'on pourra récupérer en JavaScript ou simplement de trouver tous les éléments le possédant.

récupérer toutes les `<div>` contenant les onglets. Pour les liens sur lesquels il faut cliquer pour passer d'un onglet à l'autre, ce sont juste des textes sur lesquels nous avons ajouté l'attribut `data-tab-target="#id"`.
Cet attribut a deux utilités : pouvoir récupérer facilement tous les liens en JavaScript et passer en paramètre l'`id` de la `<div>` contenant le contenu de l'onglet à afficher. Au chargement de la page, le code JavaScript récupère les liens et les `div` dans des listes et ajoute un `EventListener` à chaque lien. Le `EventListener` permet de détecter un évènement sur un élément HTML ; dans ce cas, nous ajoutons un `EventListener` de type `click` pour détecter quand l'utilisateur clique sur le lien. Quand le lien est cliqué, on récupère la valeur de `data-tab-target=""` pour avoir l'`id` de la `<div>` à afficher. On masque toutes les `<div>` avec l'attribut `data-tab-content`. On récupère la `<div>` à afficher avec un `querySelector()` et on l'affiche.

2.4.7 Mode sombre

Cette dernière fonctionnalité n'est clairement pas indispensable mais toujours agréable à avoir et ne demande pas énormément de travail de programmation, c'est pour cela que nous l'avons ajoutée. Pour rendre le développement le plus simple possible, nous avons créé un dictionnaire contenant toutes les couleurs utilisées sur le site. Nous avons aussi créé une méthode qui prend un paramètre pour savoir s'il faut afficher le site en mode sombre ou en mode clair et qui change les valeurs du dictionnaire en conséquence. Ensuite, un style spécifique est appliqué aux éléments HTML afin qu'ils s'affichent de la bonne couleur. Pour l'affichage de la simulation, les couleurs sont juste utilisées dans la fonction qui affiche une frame. Pour améliorer l'expérience utilisateur, nous avons fait appel à `window.matchMedia('(prefers-color-scheme: dark)')`. Cette fonction retourne "True" si le mode sombre est activé dans les réglages de l'appareil de l'utilisateur et "False" si c'est le mode clair qui est activé. Cela nous permet d'automatiquement choisir le même mode d'affichage que celui choisi par l'utilisateur sur son appareil. Nous avons aussi ajouté un bouton permettant de changer manuellement entre le mode sombre et le mode clair.

Chapitre 3

Discussion sur le projet final

Bien que notre simulation soit vraisemblable, il nous faut également prouver son exactitude physique. Pour ce faire, nous avons à notre disposition plusieurs outils. Il nous faut également pouvoir expliquer les cas dysfonctionnels ainsi que les problèmes rencontrés.

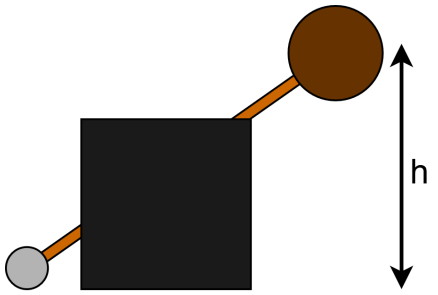
3.1 Comparaison avec d'autres modèles

3.1.1 Modèle avec conversion parfaite d'énergie potentielle à cinétique

Afin de vérifier notre trébuchet, nous pouvons énoncer un bilan d'énergie simplifié, ce qui nous permet de majorer la portée théorique maximale R et d'estimer la vitesse de lancé v_0. Pour cette estimation, nous considérons que le trébuchet ne subit pas de forces dissipatives (force de trainée, effet Magnus etc.) et que donc, l'énergie mécanique est conservée.

$$E_{mec_i} = E_{mec_f}$$

Nous considérons également que toute l'énergie du contrepoids, sous forme d'énergie potentielle, est transférée au projectile en énergie cinétique, cela signifie que le trébuchet est immobile lorsque le projectile est parti. En somme, c'est comme si l'on simplifiait notre modèle au maximum. À noter que cette méthode de vérification provient de l'une de nos sources [33].



Nous négligeons la masse du balancier par soucis de simplification.

$$\begin{aligned} \Delta E_{sys} &= (U_{m_1} - U_{m_2}) + (K_{m_1} - K_{m_2}) = 0 \\ U_{m_1} &= K_{m_2} \\ m_1 g h &= \frac{1}{2} m_2 v_0^2 \end{aligned}$$

Grâce à la 2^e loi de Newton, nous pouvons déterminer les équations du mouvement parabolique du projectile :

$$\begin{cases} x(t) = v_0 \sin\theta \cdot t \\ y(t) = -\frac{1}{2} g \cdot t^2 + v_0 \cos\theta \cdot t \end{cases}$$

Cela nous permet de déduire la portée théorique R en substituant t lorsque $y(t) = 0$.

$$R = 2\sin\theta\cos\theta \cdot \frac{v_0^2}{g} \tag{3.1}$$

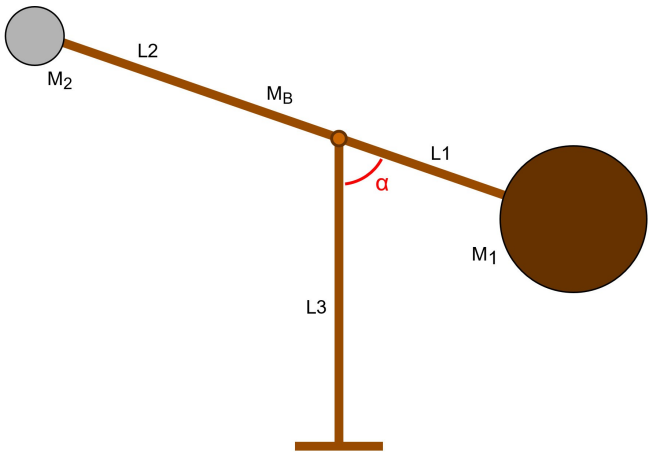
La portée maximale vaut $\frac{v_0^2}{g}$ lorsque l'angle de lancer $\theta = \frac{\pi}{2}$, ce qui est cohérent avec un modèle balistique standard.
En utilisant le bilan d'énergie énoncé plus tôt, nous pouvons déterminer un majorant pour la portée :

$$R_{max} = \frac{v_0^2}{g} = 2\frac{m_1}{m_2} \cdot h \approx 2\frac{m_1}{m_2} \cdot l_3 = 2\frac{2000}{15} \cdot 4 \approx 1067\ [m]$$

En considérant l_3 plutôt que h, on obtient un majorant beaucoup trop grand pour pouvoir nous renseigner sur la véracité de notre modèle. Il est en effet difficile de concevoir qu'un trébuchet de quatre mètres de hauteur puisse envoyer un boulet à plus d'un kilomètre. En revanche, la formule 3.1, elle, est très satisfaisante dans son approximation de la portée, en tout cas lorsque les forces dissipatives sont faibles.

3.1.2 Solution analytique d'un modèle simplifié

Nous pouvons vérifier notre intégration numérique en utilisant un modèle simplifié de notre trébuchet, s'approchant plus d'une balance. En effet, un modèle comme celui-ci possède une solution analytique et nous pourrons la comparer avec les graphes fournis par nos méthodes d'intégration.



En étudiant les moment de force et d'inertie de ce modèle, nous arrivons à l'équation différentielle $-c \cdot \sin\alpha = \ddot{\alpha}$, où c dépend de l'accélération g, des masses et des longueurs du balancier. Cette équation est cependant complexe à résoudre, c'est pourquoi nous pouvons la simplifier en approximant la fonction sinus lorsque α est petit.

$$-c \cdot \sin\alpha = \ddot{\alpha} \Rightarrow -c \cdot \alpha = \ddot{\alpha}, -\frac{\pi}{6} < \alpha < \frac{\pi}{6}$$

La solution de cette équation simplifiée est une fonction harmonique : $S(t) = A \cdot \sin(\omega \cdot t + \phi)$, où A, ω et ϕ déterminent l'amplitude, la fréquence et la constante de phase de $S(t)$.
Effectivement, notre solution est confirmée en étudiant le graphe de α lorsque la fronde et la longueur du contrepoids tendent vers zéro :

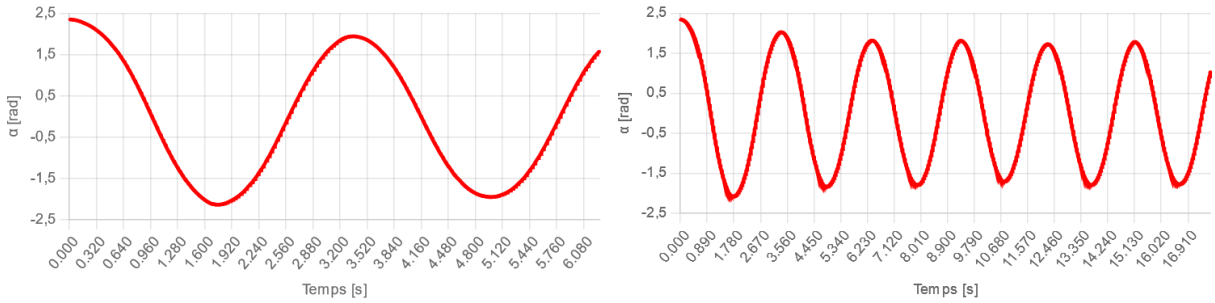


Figure 5: Graphique de α avec la fronde et le contrepoids valant 0.01 [m]

La diminution de l'amplitude au cours du temps, qui n'apparaît pas dans la solution de notre équation, est bien correcte. En effet, l'approximation du sinus que nous avons appliquée induit une légère extrapolation de la fonction. La période demeure constante, ce qui nous indique que l'intégration numérique est stable.

3.1.3 Modèle physique réduit

Pour son travail de Maturité, notre collègue Piotr Maleika a construit une maquette de trébuchet pouvant effectuer de vrais tirs. Il est donc intéressant de comparer la portée réelle d'un trébuchet avec la portée que nous pourrions obtenir à l'aide de la simulation.

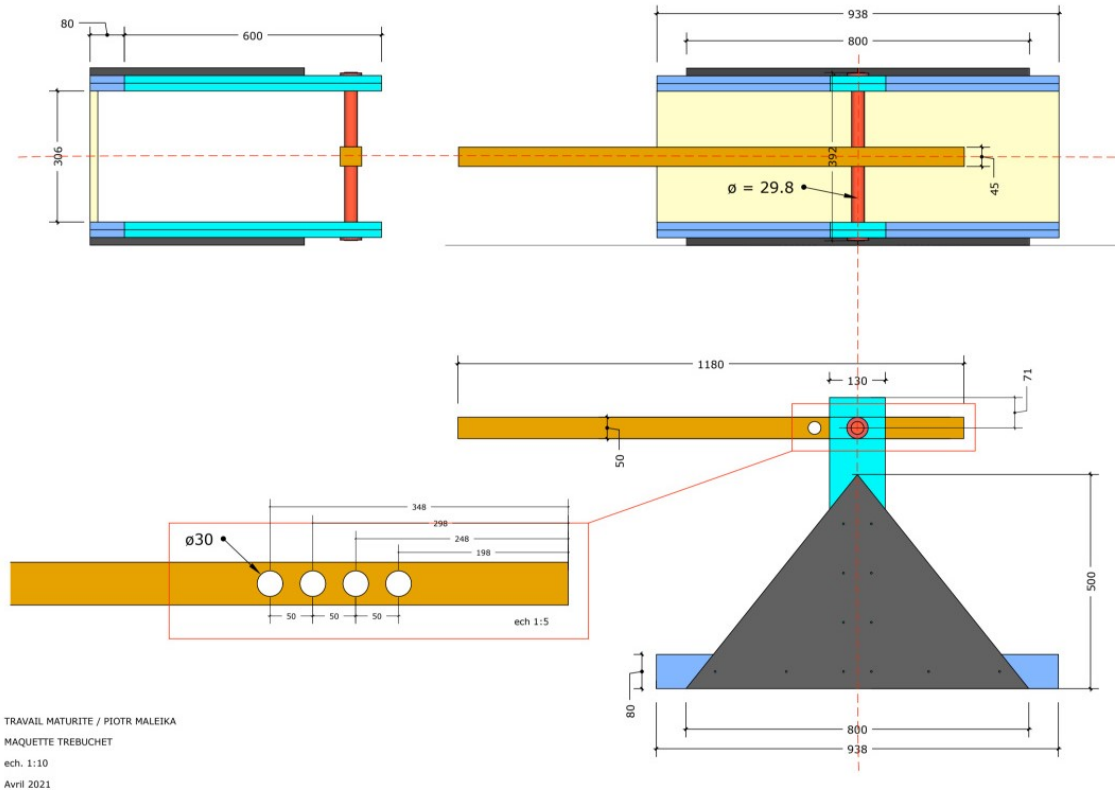


Figure 6: Plans du modèle réduit, Piotr MALEIKA

En reprenant les dimensions de son modèle, nous arrivons à ce tableau :

Données		
m_1	0.121	[kg]
m_2	16.1	[kg]
m_b	5.00	[kg]
l_1	0.298	[m]
l_2	0.882	[m]
l_3	0.529	[m]
l_4	0.250	[m]
l_5	0.980	[m]
l_b	0.590	[m]
angle de libération	≈ 40	[°]
portée JavaScript	26.0	[m]
portée physique	26.1	[m]

L'angle de libération est approximé car l'angle de lancer du modèle physique est assez compliqué à déterminer. Cependant, on peut véritablement observer la proximité entre les résultats du modèle réduit (26,1 [m]) et ceux de notre simulation (26,0 [m]).

3.2 Vérification de la simulation

3.2.1 Étude des graphiques de α, β et γ

Comme énoncé précédemment, les angles du trébuchet seuls nous permettent de décrire la dynamique de l'engin au cours du temps. C'est pourquoi, pour valider la simulation, il convient d'étudier les graphiques des angles.
En effet, puisque les équations différentielles auxquelles nous faisons face n'ont pas de solution analytique, nous devons l'approximer à l'aide d'une intégration numérique. Or, cette intégration a tendance à diverger au cours du temps, les valeurs angulaires devenant de plus en plus chaotiques. L'étude des graphiques de α, β, γ en fonction du temps nous permet d'observer lorsque la simulation s'éloigne de la réalité. Prenons par exemple les valeurs par défaut avec la méthode d'Euler :

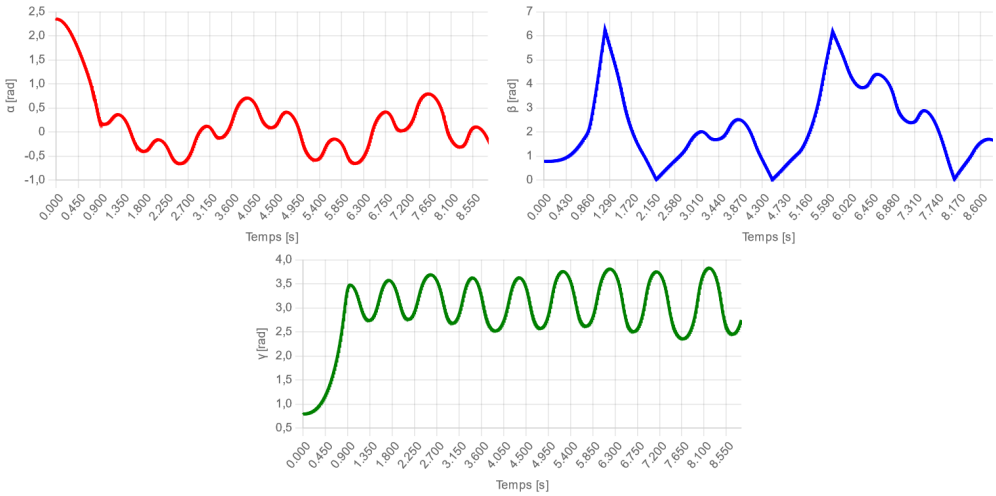


Figure 7: Graphiques des angles du trébuchet avec la méthode d'Euler

On peut déjà remarquer les "pics" de β, qui correspondent à la fronde rebondissant contre le bras de levier. Nous pouvons également constater que l'amplitude d'oscillation de γ augmente

au cours du temps, ce qui nous indique que la simulation s'éloigne de la réalité, puisque l'énergie du système ne devrait, en principe, pas augmenter.
Maintenant, observons les angles au cours du temps avec Heun :

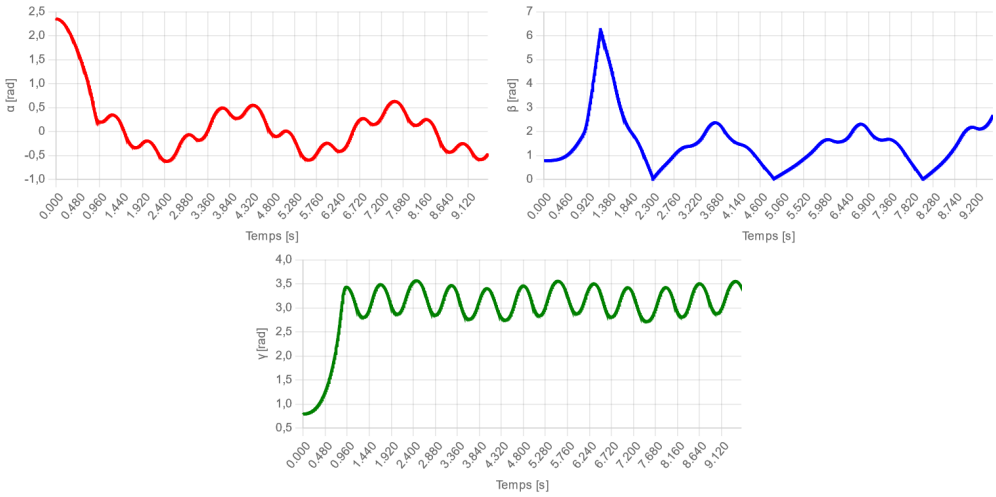


Figure 8: Graphiques des angles avec la méthode de Heun

On note clairement une différence dans la stabilité du système, l'oscillation reste constante. De plus, on peut constater la périodicité des fonctions angulaires après que le projectile a été envoyé.
Enfin, les mêmes conditions initiales, cette fois-ci avec Runge-Kutta d'ordre 4 :

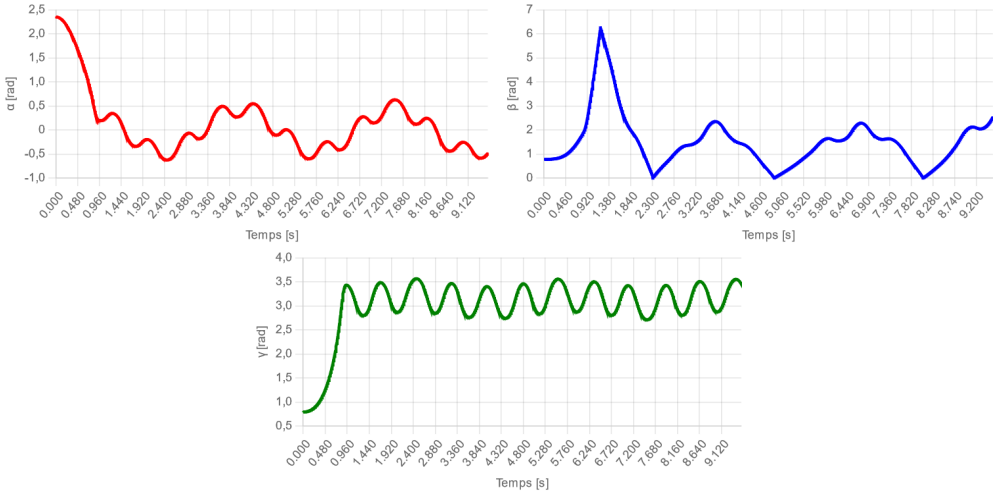


Figure 9: Graphiques des angles avec la méthode RK4

Pas de différence significative avec la méthode de Heun ; le modèle est stable au cours du temps.

3.2.2 Vérification de la conservation de l'énergie

La vérification de la conservation de l'énergie consiste à établir si l'énergie du système reste constante au cours du temps. Dans le cas inverse, cela signifierait que le système "trébuchet + projectile" subit des forces dissipatives ou bien que les calculs liés au système sont faux. La motivation de cette vérification est donc de valider nos équations de mouvement.
Pour ce faire, nous calculons à chaque pas de notre simulation la somme de l'énergie potentielle (U) et de l'énergie cinétique (K) du système, ce qui comprend le trébuchet ainsi que le projectile. Pour le trébuchet, il nous suffit de reprendre le calcul du lagrangien et de changer le signe l'énergie

potentielle afin de connaître non pas la différence entre l'énergie potentielle et cinétique mais la somme des deux.
Voilà donc le calcul que nous effectuons pour déterminer l'énergie totale du trébuchet :

$$L = T - V = K_{treb} - U_{treb} \Rightarrow E_{treb} = K_{treb} + U_{treb} = T + V$$

$$\begin{aligned} E_{treb} = &\frac{1}{2}[m_1({l_1}^2 + {l_4}^2) + m_b{l_b}^2 - 2m_1l_1l_4\cos(\gamma) + m_2({l_2}^2 + {l_5}^2) - 2m_2l_2l_5\cos(\beta)]\dot{\alpha}^2 \\ &+\frac{1}{2}m_1{l_4}^2\dot{\gamma}^2 + \frac{1}{2}m_2{l_5}^2\dot{\beta}^2 + m_1({l_4}^2 - l_1l_4\cos(\gamma))\dot{\alpha}\dot{\gamma} + m_2(-{l_5}^2 + l_2l_5\cos(\beta))\dot{\alpha}\dot{\beta} \\ &+g(-m_1l_1 + m_2l_2 + m_bl_b)\cos(\alpha) - gm_2l_5\cos(\beta - \alpha) + gm_1l_4\cos(\gamma + \alpha) \end{aligned}$$

Lorsque le projectile est libéré, nous gardons le calcul de l'énergie du trébuchet tout en changeant la valeur m_2 et en lui attribuant une masse quasi nulle ($m_2 = 0.1$ [kg]). Puis, nous lui ajoutons la somme de l'énergie potentielle et cinétique de m_2 ($\frac{1}{2}m_2v^2 + m_2gh$). Nous stockons ensuite cette valeur dans une liste et nous l'affichons sur le graphique de l'énergie.
Cependant, pour que le calcul de l'énergie soit correct, nous devons nous assurer que le système soit bien isolé, ce qui n'est pas le cas puisqu'il est en proie à des forces dissipatives liées au frottement de l'air et à l'effet Magnus. Nous isolons donc le système en l'englobant dans un fluide de densité nulle. De plus, nous modifions également le rendement des collisions afin qu'aucune perte d'énergie ne soit engendrée en lui donnant la valeur de 1, ce qui signifie que la vitesse de rebond sera la même qu'à l'impact. Dans ces conditions, si nos calculs sont corrects, l'énergie ne devrait presque pas varier.
Toutefois, l'énergie subit toujours de petites variations, elles sont dues à l'intégration numérique ainsi qu'à l'approximation de ces collisions, cependant, ce phénomène reste minime dans des conditions normales, comme on peut le voir dans les graphiques de l'énergie en fonction du temps. Prenons par exemple deux modèles à échelles différentes. Le premier correspond aux dimensions entrées par défaut tandis que le second est le modèle réduit construit par Piotr Maleika, mesurant ≈ 1 mètre :

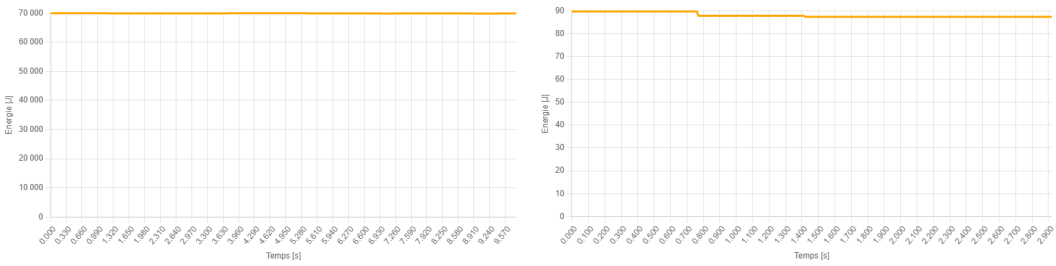


Figure 10: Énergie en fonction du temps avec collisions

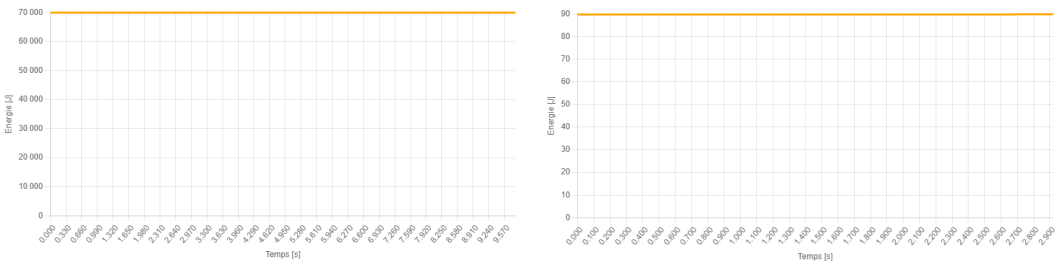


Figure 11: Énergie en fonction du temps sans collisions

Tout d'abord, en comparant le second modèle avec et sans les collisions, nous pouvons attester que celles-ci peuvent créer de légères fluctuations de l'énergie. Nous pouvons également supposer que le même phénomène se produit pour le premier modèle, toutefois on ne peut pas l'observer à cause de l'échelle. Par ces observations, nous pouvons affirmer que les collisions amènent un facteur d'erreur, cependant ce facteur reste véritablement insignifiant.
La chose la plus notable à remarquer ici, est qu'avec ou sans les collisions, l'énergie du système semble rester constante dans le temps. Cela permet de confirmer que notre système a été implémenté correctement puisqu'il respecte le principe de la conservation de l'énergie. Nous avons donc une preuve concrète de la véracité de nos calculs.

Maintenant, comparons nos calculs avec ceux provenant de l'article rédigé par Robin De Jong, de l'Université de Delft [21]. Nous avons donc remplacé nos coefficients de la matrice des équations du trébuchet par les leurs afin de déterminer lequel de nos deux projet contient des erreurs dans la dérivation des équations, sachant que le Lagrangien est le même. Voyons l'énergie au cours du temps avec les dimensions ordinaires et le modèle réduit en utilisant les coefficients de Robin de Jong :

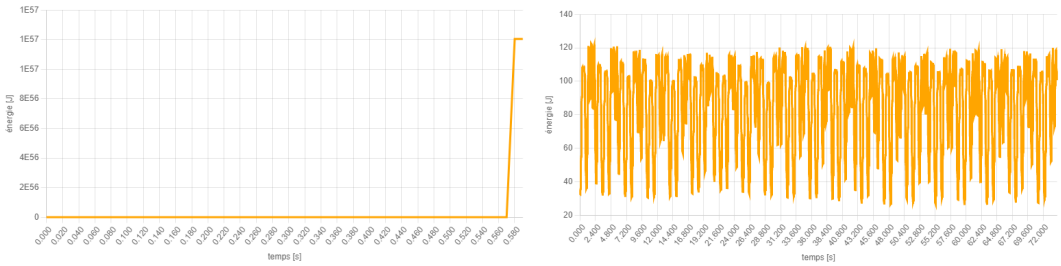


Figure 12: Énergies de deux modèles avec collisions, en utilisant les coefficients de Delft

En comparant les graphiques ci-dessus, l'on voit bel et bien que l'énergie n'est pas conservée dans le temps. En addition de cela, le graphique de l'énergie du second modèle est purement chaotique. Ces deux problèmes indiquent bel et bien une erreur dans le système. Après vérification de leurs calculs, et en les refaisant avec le même Lagrangien, nous avons pu conclure que leurs coefficients [1] étaient erronés et non les nôtres.

3.3 Problèmes rencontrés

3.3.1 Temps de calcul trop long

Comme décrit précédemment, les calculs des positions et des angles sont stockés dans des listes, qui sont ensuite lues par le programme. Cependant, dans le cas où les calculs de l'intégration numérique sont trop long à effectuer, par exemple si la vitesse du vent est trop importante, le programme ne pourra pas remplir ces listes dans un temps réduit, ce qui fera geler la page du site ainsi que le navigateur.

3.3.2 Problème d'inversion de matrice

Dans la plupart des cas, les problèmes sont dus à l'inversion de la matrice des coefficients. Le programme doit d'abord vérifier que le déterminant de cette matrice n'est pas égal à zéro, car par définition, on ne peut pas inverser une matrice dont le déterminant vaut zéro [2]. Or, si

1. Pour être plus précis, l'erreur se situe dans le coefficient w_3
2. Tout comme on ne peut pas se déplacer sur un plan en utilisant une combinaison linéaire de vecteurs colinéaires.

le déterminant de cette matrice tend vers zéro, ce qui peut arriver lorsque les valeurs entrées sont aberrantes, les accélérations angulaires qui ressortent peuvent être étonnamment grandes, ce qui va encore s'extrapoler avec l'intégration numérique. À la sortie, les angles augmentent et peuvent atteindre plusieurs centaines de radians, ce qui est réellement impossible. Ce type d'erreur arrive surtout lorsque les masses tendent vers zéro ou bien lorsque les longueurs du trébuchet sont inhabituelles.

3.3.3 Violation du domaine couvert (`Overflow`)

Un `Overflow` se produit lorsque le résultat d'une opération mathématique dans un programme dépasse le domaine couvert sur nombre de bits prévu à cet effet, ce qui invalide le résultat. Par exemple, si sur trois bits nous souhaitons additionner en binaire 7 (111) et 1 (001) le résultat serait 1000, cependant, puisque le résultat ne peut être stocké que sur 3 bits, le premier bit ne sera pas enregistré et le résultat de 7 + 1 donné par l'ordinateur sera 0. Ce problème reste très théorique dans notre cas car les nombres en JavaScript sont enregistrés avec la norme IEEE 754 double précision (sur 64 bits). Cette norme nous permet d'effectuer des opérations binaires sur des nombres rationnels. Le plus grand nombre positif et le plus petit nombre négatif possible avec cette norme sont $\pm(2^{1024} - 2^{971}) \approx \pm 1,798 \cdot 10^{308}$ [20]. Il est à noter que lorsque qu'une valeur dépasse le domaine couvert en JavaScript, il n'est pas affiché la valeur `NaN`, comme dans la plupart des langage de programmation, mais la valeur `Infinity`. Celle-ci a la particularité d'encore fonctionner avec certaines opérations.
Même s'il est très peu probable qu'une opération donne un résultat qui viole le domaine couvert, nous tenions à l'aborder ici, car ce problème peut être bien réel dans certains cas. L'exemple le plus connu de ce bogue est sûrement l'explosion du vol 501 de la fusée Ariane 5 en 1996, causée par un overflow dans le programme de guidage du lanceur [9].
Dans le cadre de notre programme, plusieurs facteurs peuvent causer un overflow. Il peut être provoqué par un rapport trop faible entre la masse de la barre et la masse du contrepoids ($\frac{m_b}{m_1}$). En effet, après le lancer, il ne reste généralement dans des conditions normales qu'une masse très faible pour la fronde ainsi que la masse de la barre pour contrer la masse du contrepoids. Dans le cas où la masse de la barre est trop faible, les accélérations angulaires deviennent très grandes, ce qui rend les approximations de l'intégration numérique de plus en plus mauvaises et peut finalement conduire à une augmentation trop rapides des vitesses angulaires et des angles. La finalité de ceci est que ces valeurs atteignent de telles grandeurs que des `Infinity` apparaissent. Ces `Infinity`, utilisés dans les fonctions trigonométriques se transforment en `NaN`. La fonction chargée alors d'afficher le trébuchet ne sait que faire de ces valeurs. Il en résulte donc finalement un mauvais affichage de la simulation du trébuchet, voire l'arrêt complet de celle-ci.

Conclusion

C'est avec satisfaction que nous rendons ce projet qui, au début, nous paraissait pourtant insurmontable. Malgré quelques cas dysfonctionnels, notre simulation est en effet largement satisfaisante, en particulier au vu de notre niveau d'études ainsi que de la nouveauté des outils mathématiques et physiques utilisés. Elle parvient à afficher avec précision la trajectoire d'un projectile lancé avec un trébuchet en tenant compte de nombreux paramètres, c'est pourquoi nous estimons que le but du travail est atteint. Un développement complémentaire pourrait être de implémenter la simulation en trois dimensions, en utilisant des outils comme Unity ou UnrealEngine.

De cette expérience, nous avons tiré des compétences de recherche et de résolution de problèmes, à travers le tri des sources et la lecture d'articles scientifiques. Nous avons également acquis des connaissances sur la programmation en JavaScript, sur la dynamique des fluide et sur les équations de Lagrange, ce qui nous sera utile dans la suite de nos cursus. Cependant, la chose la plus précieuse que nous retirons de ces dix mois de travail est sans doute la capacité à s'organiser pour un projet de cette ampleur.

En outre, le fait d'observer que les outils mathématiques purs ne sont pas assez puissants pour déterminer parfaitement des modèles physiques comme le trébuchet nous rappelle humblement la difficulté, pour l'humain, de retranscrire la réalité à travers la Science.

Table des figures

Bibliographie

Livres

[4] Comissions romandes de mathématique (CRM). *Formulaires et tables*. p. 101. CRM Éditions, 2015. ISBN : 9782940621057.

[22] René Lafrance. *Physique 1 - Mécanique*. De Boeck Supérieur, 2015. ISBN : 9782804190682.

Articles et sources universitaires

[7] Habib Ammari. *Lecture 4 : Numerical solution of ordinary differential equations*. Eidgenössische Technische Hochschule (ETH) Zürich. URL : http://www.sam.math.ethz.ch/~grsam/SS20/NAII/resources/slides/ODE-Lecture4.pdf (visité le 20/10/2021).

[9] Chris Baraniuk. BBC, 2015. URL : https://www.bbc.com/future/article/20150505-the-numbers-that-lead-to-disaster (visité le 20/10/2021).

[18] W. G. Harter. *Unit 2 Lagrangian and Hamiltonian Mechanics*. University of Arkansas. URL : https://pirelli.hosted.uark.edu/testing/Unit2/CMwBang!10.htm (visité le 19/10/2021).

[21] Robin de Jong. *Modelling the movements of a counterweight trebuchet while firing*. Sous la dir. de B. J. Meulenbroek. Technische Universiteit (TU) Delft, 2020. URL : https://repository.tudelft.nl/islandora/object/uuid:9c66e4d3-d048-4d1b-8234-89664682181f/datastream/OBJ/download (visité le 17/10/2021).

[24] Stephen Lucas. *What affects the range of a trebuchet ?* University College London (UCL). URL : https://www.ucl.ac.uk/~zcapf71/Trebuchet%20coursework%20for%2website.pdf (visité le 19/10/2021).

[27] Aaron Mosher. *A Mathematical Model for a Trebuchet*. Washington University in St. Louis. URL : https://classes.engineering.wustl.edu/2009/fall/ese251/presentations/(AAM_13)Trebuchet.pdf (visité le 19/10/2021).

[28] Jason T. Necaise. *Trebuchet Optimization*. Massaschusetts Institute of Technology (MIT), 2019. URL : http://necaise.mit.edu/sites/default/files/documentsJason_Necaise_Trebuchet_Project.pdf (visité le 17/10/2021).

[31] D. Pastre. *chapitre 2 - Méthode de Gauss-Jordan - Calcul de l'inverse d'une matrice*. Université René Descartes, Paris V, 2003. URL : http://www.normalesup.org/~pastre/meth-num/MN/2-gauss-jordan/cours-gauss-jordan.pdf (visité le 20/10/2021).

[33] Donald B. Siano. *Trebuchet Mechanics*. Weber State University (WSU), Utah, 2013. URL : http://www.algobeautytreb.com/trebmath356.pdf (visité le 17/10/2021).

[42] Vincent Vuille. *Algèbre linéaire*. Gymnase de Renens (GyRe), 2019. URL : https://drive.google.com/file/d/1WbiwFVtmTHtraT-fmQmuN0W2T-lMinNP/view (visité le 20/10/2021).

[44] Jürgen Weigert. *Ballistik einer Mittelalterlichen Kriegsmaschine*. Friedrich-Alexander-Universität (FAU) Erlangen-Nürnberg, 2011. URL : https://en.opensuse.org/images/2/20/Trebuchet_simulation_slides.pdf (visité le 19/10/2021).

Sources Internet

[1] C++ Community. 2016. URL : https://www.c-plusplus.net/forum/topic/338297/matrix-invertieren-c-mit-algorithmus/7 (visité le 18/10/2021).

[2] URL : https://brm.io/matter-js/demo/#catapult (visité le 19/10/2021).

[3] URL : http://www.benchtophybrid.com/How_to_Simulate_a_Trebuchet_Part1.pdf (visité le 19/10/2021).

[5] *8. Lagrangian dynamics*. 2020. URL : https://def.fe.up.pt/dynamics/lagrangian_dynamics.html (visité le 19/10/2021).

[6] *9 - Euler's Numerical Method*. University of Alabama in Huntsville. URL : https://www.uah.edu/images/people/faculty/howellkb/DEText-Ch9.pdf (visité le 20/10/2021).

[8] Alexandre Astier. *L'étudiant*. Kaamelott - Livre III - Épisode 95. CALT. URL : https://youtu.be/eRyF_yckoJ4 (visité le 20/10/2021).

[10] Luis Batalha. *Double Pendulum - Double Pendulum in less than 100 lines of JavaScript*. URL : http://www.physicsandbox.com/projects/double-pendulum.html (visité le 20/10/2021).

[11] Renaud Beffeyte. *Trébuchets / Biffa / Bride / Brède*. 2011. URL : http://medieval.mrugala.net/Armes%20de%20siege/Trebuchet.htm (visité le 19/10/2021).

[12] *Chapter 2 - Lagrange's and Hamilton's Equations*. Rutgers, The State University of New Jersey. URL : https://www.physics.rutgers.edu/~shapiro/507/book3.pdf (visité le 19/10/2021).

[13] *Double Pendulum*. URL : https://www.myphysicslab.com/pendulum/double-pendulum-en.html (visité le 20/10/2021).

[14] MOOC Mécanique EPFL. *25.2 Méthode de Lagrange, applications*. École Polytechnique Fédérale de Lausanne (EPFL). 2014. URL : https://youtu.be/G7phtTMEUII (visité le 19/10/2021).

[15] *Euler method*. Wikipedia, the free encyclopedia. URL : https://en.wikipedia.org/wiki/Euler_method (visité le 17/10/2021).

[16] Gilbert Gastebois. *L'effet Magnus*. URL : http://gilbert.gastebois.pagesperso-orange.fr/java/magnus/Magnus_theorie.pdf (visité le 17/10/2021).

[17] *Gaussian elimination*. Wikipedia, the free encyclopedia. URL : https://en.wikipedia.org/wiki/Gaussian_elimination (visité le 17/10/2021).

[19] *Heun's method*. Wikipedia, the free encyclopedia. URL : https://en.wikipedia.org/wiki/Heun%27s_method (visité le 17/10/2021).

[20] *IEEE 754*. Wikipédia, l'encyclopédie libre. URL : https://fr.wikipedia.org/wiki/IEEE_754 (visité le 21/10/2021).

[23] *Lagrange's Equations*. Massaschusetts Institute of Technology (MIT). 2003. URL : https://ocw.mit.edu/courses/aeronautics-and-astronautics/16-61-aerospace-dynamics-spring-2003/lecture-notes/lecture7.pdf (visité le 19/10/2021).

[25] *Méthodes de Runge-Kutta*. Wikipédia, l'encyclopédie libre. URL : https://fr.wikipedia.org/wiki/M%C3%A9thodes_de_Runge-Kutta (visité le 17/10/2021).

[26] Étienne MIQUEY. *Trajectoire et rebond d'une balle au tennis de table*. 2008. URL : https://www.irif.fr/~emiquey/stuff/TIPE.pdf (visité le 19/10/2021).

[29] *Numerical Solution of ODEs*. Auckland University. URL : https://www.math.auckland.ac.nz/class363/notes/ODEs.pdf (visité le 20/10/2021).

[30] *Numerical solutions to Ordinary differential equations*. URL : https://jurasic-park.de/ode/ (visité le 20/10/2021).

[32] *Poussée d'Archimède*. Wikipédia, l'encyclopédie libre. URL : https://fr.wikipedia.org/wiki/Pouss%C3%A9e_d%27Archim%C3%A8de (visité le 18/10/2021).

[34] Enseignement SUPÉRIEUR. *FORMALISME LAGRANGIEN*. Université Paris-Saclay. 2020. URL : https://www.youtube.com/watch?v=H_feeQeqOlQ (visité le 19/10/2021).

[35] *The double pendulum*. University of California. 2021. URL : https://rotations.berkeley.edu/the-double-pendulum/ (visité le 20/10/2021).

[36] *Traînée*. Wikipédia, l'encyclopédie libre. URL : https://fr.wikipedia.org/wiki/Tra%C3%AEn%C3%A9e (visité le 18/10/2021).

[37] *Trebuchet*. Wikipedia, the free encyclopedia. URL : https://en.wikipedia.org/wiki/Trebuchet (visité le 17/10/2021).

[38] *Trébuchet*. Wikipédia, l'encyclopédie libre. URL : https://fr.wikipedia.org/wiki/Tr%C3%A9buchet (visité le 17/10/2021).

[39] *Trebuchet Physics*. URL : https://www.real-world-physics-problems.com/trebuchet-physics.html (visité le 19/10/2021).

[40] *Trebuchet : The Dynamics of a Medieval Siege Engine*. URL : http://www.uphysicsc.com/2010-GM-B-210.pdf (visité le 19/10/2021).

[41] *VirtualTrebuchet*. URL : http://www.virtualtrebuchet.com/.

[43] Vincent VUILLE. *Projet : trajectoire d'une balle de tennis*. Gymnase de Renens (GyRe). 2018. URL : https://drive.google.com/file/d/11eDmGNOmB2uK9mFH93TrsS-6k5HdKaqF/view (visité le 19/10/2021).

Annexe A

Développement des calculs

A.1 Calcul du Lagrangien

Voici les calculs du Lagrangien L du système ainsi que des équations de Lagrange qui en découlent. Pour déterminer le Lagrangien, il nous faut connaître les énergies cinétique T et potentielle V du système.

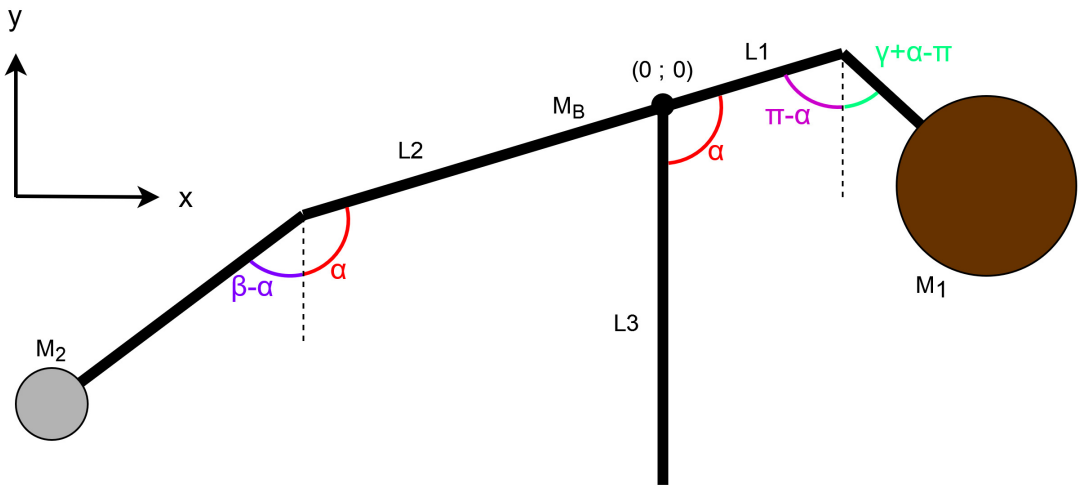


Figure 13: Trébuchet avec le référentiel et les angles déduits

A.1.1 Calcul de l'énergie cinétique T

Nous commençons par calculer le carré des vitesses angulaires au pour chaque masse, cela nous sera utile pour ensuite calculer l'énergie cinétique. Ces vitesses carrées sont v_4^2 pour le contrepoids, v_5^2 pour la masse du projectile et enfin v_b^2 pour la masse de la barre.

$v_4{}^2 = (l_1\dot{\alpha} + l_4(\dot{\gamma}+\dot{\alpha})\cdot\cos(\pi-\gamma))^2 + (l_4(\dot{\gamma}+\dot{\alpha})\cdot\sin(\pi-\gamma))^2 = (l_1\dot{\alpha} + l_4(\dot{\gamma}+\dot{\alpha})\cdot(-\cos(\gamma)))^2 + (l_4(\dot{\gamma}+\dot{\alpha})\cdot\sin(\gamma))^2 = l_1{}^2\dot{\alpha}^2 + l_4{}^2(\dot{\gamma}+\dot{\alpha})^2 - 2l_1l_4\dot{\alpha}(\dot{\gamma}+\dot{\alpha})\cdot\cos(\gamma) = \left(l_1{}^2 + l_4{}^2 - 2l_1l_4\cdot\cos(\gamma)\right)\dot{\alpha}^2 + l_4{}^2\dot{\gamma}^2 + \left(2l_4^2 - 2l_1l_4\cos(\gamma)\right)\dot{\alpha}\dot{\gamma}$

$v_5{}^2 = \left(-l_2\dot{\alpha} + l_5(\dot{\beta}-\dot{\alpha})\cdot\cos(\pi-\beta)\right)^2 + \left(l_5(\dot{\beta}-\dot{\alpha})\cdot\sin(\pi-\beta)\right)^2 = \left(-l_2\dot{\alpha} + l_5(\dot{\beta}-\dot{\alpha})\cdot(-\cos(\beta))\right)^2 + \left(l_5(\dot{\beta}-\dot{\alpha})\cdot\sin(\beta)\right)^2 = l_2{}^2\dot{\alpha}^2 + l_5{}^2(\dot{\beta}-\dot{\alpha})^2 + 2l_2l_5\dot{\alpha}(\dot{\beta}-\dot{\alpha})\cdot\cos(\beta) = \left(l_2{}^2 + l_5{}^2 - 2l_2l_5\cdot\cos(\beta)\right)\dot{\alpha}^2 + l_5{}^2\dot{\beta}^2 + \left(-2l_5{}^2 2l_2l_5\cos(\beta)\right)\dot{\alpha}\dot{\beta}$

$v_b{}^2 = (-l_b\dot{\alpha})^2 = l_b{}^2\dot{\alpha}^2$

Avec ces vitesses au carré, nous pouvons calculer l'énergie cinétique grâce à la formule $T = \frac{1}{2}mv^2$:

$T_1 = \frac{1}{2}m_1\left[\left(l_1{}^2 + l_4{}^2 - 2l_1l_4\cdot\cos(\gamma)\right)\dot{\alpha}^2 + l_4{}^2\dot{\gamma}^2 + \left(2l_4{}^2 - 2l_1l_4\cos(\gamma)\right)\dot{\alpha}\dot{\gamma}\right]$
$T_2 = \frac{1}{2}m_2\left[\left(l_2{}^2 + l_5{}^2 - 2l_2l_5\cdot\cos(\beta)\right)\dot{\alpha}^2 + l_5{}^2\dot{\beta}^2 + \left(-2l_5{}^2 + 2l_2l_5\cos(\beta)\right)\dot{\alpha}\dot{\beta}\right]$
$T_b = \frac{1}{2}m_bl_b{}^2\dot{\alpha}^2$

$T = T_1+T_2+T_b = \frac{1}{2}\left[m_1\left(l_1{}^2 + l_4{}^2\right) + m_2\left(l_2{}^2 + l_5{}^2\right) + m_bl_b{}^2 - 2m_1l_1l_4\cos(\gamma) - 2m_2l_2l_5\cos(\beta)\right]\dot{\alpha}^2 +$
$\frac{1}{2}m_2l_5{}^2\dot{\beta}^2 + m_2\left(-l_5{}^2 + l_2l_5\cos(\beta)\right)\dot{\alpha}\dot{\beta} + \frac{1}{2}m_1l_4{}^2\dot{\gamma}^2 + m_1\left(l_4{}^2 - l_1l_4\cos(\gamma)\right)\dot{\alpha}\dot{\gamma}$

A.1.2 Calcul de l'énergie potentielle V

L'énergie potentielle gravitationelle se calcule par la formule $V = mgh$. Afin de déterminer V, il nous faut donc placer un référentiel. Nous considérons donc que l'origine du référentiel c'est-à-dire le point ayant comme coordonnées $(0;0)$, se trouve sur le pivot principal du trébuchet. Connaissant l'accélération gravitationnelle g et les masses, il nous suffit de déduire les hauteurs.

$V_b = m_bg\left(-l_b\cdot\cos(\pi-\alpha)\right) = m_bgl_b\cdot\cos(\alpha)$
$V_1 = m_1g\left(l_1\cdot\cos(\pi-\alpha) - l_4\cdot\cos(\gamma+\alpha-\pi)\right) = m_1g\left(-l_1\cdot\cos(\alpha) + l_4\cdot\cos(\gamma+\alpha)\right)$
$V_2 = m_2g\left(-l_2\cdot\cos(\pi-\alpha) - l_5\cdot\cos(\beta-\alpha)\right) = m_2g\left(l_2\cdot\cos(\alpha) - l_5\cdot\cos(\beta-\alpha)\right)$

$V = V_1 + V_2 + V_b = g\left(-m_1l_1 + m_2l_2 + m_bl_b\right)\cos(\alpha) - gm_2l_5\cos(\beta-\alpha) + gm_1l_4\cos(\gamma+\alpha)$

A.1.3 Lagrangien

Maintenant que nous connaissons l'énergie potentielle et cinétique du système. Il nous suffit de faire la différence entre l'énergie potentielle et cinétique du système pour obtenir le Lagrangien :

$L = T - V$
$= \frac{1}{2}[m_1(l_1{}^2+l_4{}^2) + m_bl_b{}^2 - 2m_1l_1l_4\cos(\gamma) + m_2(l_2{}^2+l_5{}^2) - 2m_2l_2l_5\cos(\beta)]\dot{\alpha}^2 + \frac{1}{2}m_1l_4{}^2\dot{\gamma}^2 +$
$\frac{1}{2}m_2l_5{}^2\dot{\beta}^2 + m_1(l_4{}^2 - l_1l_4\cos(\gamma))\dot{\alpha}\dot{\gamma} + m_2(-l_5{}^2 + l_2l_5\cos(\beta))\dot{\alpha}\dot{\beta} - g(-m_1l_1 + m_2l_2 +$
$m_bl_b)\cos(\alpha) + gm_2l_5\cos(\beta-\alpha) - gm_1l_4\cos(\gamma+\alpha)$

A.2 Calcul des équations de Lagrange

Commençons par les calculs de la première équation : $\frac{d}{dt}\left(\frac{\partial L}{\partial\dot{\alpha}}\right) - \frac{\partial L}{\partial\alpha} = 0$

$\frac{\partial L}{\partial\alpha} = g(-m_1l_1 + m_2l_2 + m_bl_b)\sin(\alpha) + gm_2l_5\sin(\beta-\alpha) + gm_1l_4\cos(\gamma+\alpha)$

$\frac{\partial L}{\partial\dot{\alpha}} = \left[m_1(l_1{}^2 + l_4{}^2) + m_bl_b{}^2 - 2m_1l_1l_4\cos(\gamma) + m_2(l_2{}^2 + l_5{}^2) - 2m_2l_2l_5\cos(\beta)\right]\dot{\alpha} + m_1(l_4{}^2 -$
$l_1l_4\cos(\gamma))\dot{\gamma} + m_2(-l_5{}^2 + l_2l_5\cos(\beta))\dot{\beta}$

$\frac{d}{dt}\left(\frac{\partial L}{\partial\dot{\alpha}}\right) = \left[m_1(l_1{}^2 + l_4{}^2) + m_bl_b{}^2 - 2m_1l_1l_4\cos(\gamma) + m_2(l_2{}^2 + l_5{}^2) - 2m_2l_2l_5\cos(\beta)\right]\ddot{\alpha} +$
$[2m_2l_2l_5\dot{\beta}\sin(\beta) + 2m_1l_1l_4\dot{\gamma}\sin(\gamma)]\dot{\alpha} + m_1(l_4{}^2 - l_1l_4\cos(\gamma))\ddot{\gamma} + m_1l_1l_4\sin(\gamma)\dot{\gamma}^2 +$
$m_2\left(-l_5{}^2 + l_2l_5\cos(\beta)\right)\ddot{\beta} - m_2l_2l_5\sin(\beta)\dot{\beta}^2$

$\frac{d}{dt}\left(\frac{\partial L}{\partial\dot{\alpha}}\right) - \frac{\partial L}{\partial\alpha} = \left[m_1(l_1{}^2 + l_4{}^2) + m_bl_b{}^2 - 2m_1l_1l_4\cos(\gamma) + m_2(l_2{}^2 + l_5{}^2) - 2m_2l_2l_5\cos(\beta)\right]\ddot{\alpha} +$
$m_2(-l_5{}^2 + l_2l_5\cos(\beta))\ddot{\beta} + m_1(l_4{}^2 - l_1l_4\cos(\gamma))\ddot{\gamma} + (2m_2l_2l_5\dot{\beta}\sin(\beta) + 2m_1l_1l_4\dot{\gamma}\sin(\gamma))\dot{\alpha} +$
$m_1l_1l_4\sin(\gamma)\dot{\gamma}^2 - m_2l_2l_5\sin(\beta)\dot{\beta}^2 - g(-m_1l_1 + m_2l_2 + m_bl_b)\sin(\alpha) - gm_2l_5\sin(\beta-\alpha) -$
$gm_1l_4\sin(\gamma+\alpha) = 0$

Ensuite la seconde équation : $\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\beta}}\right) - \frac{\partial L}{\partial \beta} = 0$

$\frac{\partial L}{\partial \beta} = m_2 l_2 l_5 \sin(\beta)\dot{\alpha}^2 - m_2 l_2 l_5 \sin(\beta)\dot{\alpha}\dot{\beta} - g m_2 l_5 \sin(\beta - \alpha)$

$\frac{\partial L}{\partial \dot{\beta}} = m_2 {l_5}^2 \dot{\beta} + m_2(-{l_5}^2 + l_2 l_5 \cos(\beta))\dot{\alpha}$

$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\beta}}\right) = m_2 {l_5}^2 \ddot{\beta} + m_2(-{l_5}^2 + l_2 l_5 \cos(\beta))\ddot{\alpha} - m_2 l_2 l_5 \sin(\beta)\dot{\alpha}\dot{\beta}$

$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\beta}}\right) - \frac{\partial L}{\partial \beta} = m_2 {l_5}^2 \ddot{\beta} + m_2(-{l_5}^2 + l_2 l_5 \cos(\beta))\ddot{\alpha} - m_2 l_2 l_5 \sin(\beta)\dot{\alpha}\dot{\beta} - m_2 l_2 l_5 \sin(\beta)\dot{\alpha}^2 + m_2 l_2 l_5 \sin(\beta)\dot{\alpha}\dot{\beta} + g m_2 l_5 \sin(\beta - \alpha) = m_2(-{l_5}^2 + l_2 l_5 \cos(\beta))\ddot{\alpha} + m_2 {l_5}^2 \ddot{\beta} - m_2 l_2 l_5 \sin(\beta)\dot{\alpha}^2 + g m_2 l_5 \sin(\beta - \alpha) = 0$

Finalement la troisième équation : $\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\gamma}}\right) - \frac{\partial L}{\partial \gamma} = 0$

$\frac{\partial L}{\partial \gamma} = m_1 l_1 l_4 \sin(\gamma)\dot{\alpha}^2 + m_1 l_1 l_4 \sin(\gamma)\dot{\alpha}\dot{\gamma} + g m_1 l_4 \sin(\gamma + \alpha)$

$\frac{\partial L}{\partial \dot{\gamma}} = m_1 {l_4}^2 \dot{\gamma} + m_1({l_4}^2 - l_1 l_4 \cos(\gamma))\dot{\alpha}$

$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\gamma}}\right) = m_1 {l_4}^2 \ddot{\gamma} + m_1({l_4}^2 - l_1 l_4 \cos(\gamma))\ddot{\alpha} + m_1 l_1 l_4 \sin(\gamma)\dot{\alpha}\dot{\gamma}$

$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\gamma}}\right) - \frac{\partial L}{\partial \gamma} = m_1 {l_4}^2 \ddot{\gamma} + m_1({l_4}^2 - l_1 l_4 \cos(\gamma))\ddot{\alpha} + m_1 l_1 l_4 \sin(\gamma)\dot{\alpha}\dot{\gamma} - m_1 l_1 l_4 \sin(\gamma)\dot{\alpha}^2 - m_1 l_1 l_4 \sin(\gamma)\dot{\alpha}\dot{\gamma} - g m_1 l_4 \sin(\gamma + \alpha) = m_1({l_4}^2 - l_1 l_4 \cos(\gamma))\ddot{\alpha} + m_1 {l_4}^2 \ddot{\gamma} - m_1 l_1 l_4 \sin(\gamma)\dot{\alpha}^2 - g m_1 l_4 \sin(\gamma + \alpha) = 0$

A.3 Application des différentes méthodes d'analyse numérique sur le système

A.3.1 Mouvement du trébuchet

Méthode d'Euler

$$t_{n+1} = t_n + h$$

$$p_n = calculsAccAngu(\alpha_n, \beta_n, \gamma_n, \dot{\alpha}_n, \dot{\beta}_n, \dot{\gamma}_n)$$

$$\dot{\alpha}_{n+1} = \dot{\alpha}_n + p_{n;\alpha} \cdot h$$

$$\dot{\beta}_{n+1} = \dot{\beta}_n + p_{n;\beta} \cdot h$$

$$\dot{\gamma}_{n+1} = \dot{\gamma}_n + p_{n;\gamma} \cdot h$$

$$\alpha_{n+1} = \alpha_n + \dot{\alpha}_n \cdot h + \frac{1}{2} \cdot p_{n;\alpha} \cdot h^2$$

$$\beta_{n+1} = \beta_n + \dot{\beta}_n \cdot h + \frac{1}{2} \cdot p_{n;\beta} \cdot h^2$$

$$\gamma_{n+1} = \gamma_n + \dot{\gamma}_n \cdot h + \frac{1}{2} \cdot p_{n;\gamma} \cdot h^2$$

Méthode de Heun

$$t_{n+1} = t_n \ + \ h$$

$$p_n = calculsAccAngu(\alpha_n, \beta_n, \gamma_n, \dot{\alpha}_n, \dot{\beta}_n, \dot{\gamma}_n)$$

$$q_n = calculsAccAngu(\alpha_n \ + \ \dot{\alpha}_n \cdot h \ + \ \tfrac{1}{2} \cdot p_{n;\alpha} \cdot h^2, \beta_n \ + \ \dot{\beta}_n \cdot h \ + \ \tfrac{1}{2} \cdot p_{n;\beta} \cdot h^2, \gamma_n \ + \ \dot{\gamma}_n \cdot h \ + \ \tfrac{1}{2} \cdot p_{n;\gamma} \cdot h^2, \dot{\alpha}_n \ + \ p_{n;\alpha} \cdot h, \dot{\beta}_n \ + \ p_{n;\beta} \cdot h, \dot{\gamma}_n \ + \ p_{n;\gamma} \cdot h)$$

$$\dot{\alpha}_{n+1} = \dot{\alpha}_n \ + \ \frac{p_{n;\alpha} \ + \ q_{n;\alpha}}{2} \cdot h$$

$$\dot{\beta}_{n+1} = \dot{\beta}_n \ + \ \frac{p_{n;\beta} \ + \ q_{n;\beta}}{2} \cdot h$$

$$\dot{\gamma}_{n+1} = \dot{\gamma}_n \ + \ \frac{p_{n;\gamma} \ + \ q_{n;\gamma}}{2} \cdot h$$

$$\alpha_{n+1} = \alpha_n \ + \ \dot{\alpha}_n \cdot h \ + \ \frac{1}{2} \cdot \frac{p_{n;\alpha} \ + \ q_{n;\alpha}}{2} \cdot h^2$$

$$\beta_{n+1} = \beta_n \ + \ \dot{\beta}_n \cdot h \ + \ \frac{1}{2} \cdot \frac{p_{n;\beta} \ + \ q_{n;\beta}}{2} \cdot h^2$$

$$\gamma_{n+1} = \gamma_n \ + \ \dot{\gamma}_n \cdot h \ + \ \frac{1}{2} \cdot \frac{p_{n;\gamma} \ + \ q_{n;\gamma}}{2} \cdot h^2$$

Méthode de Runge - Kutta (ordre 4)

$$t_{n+1} = t_n \ + \ h$$

$$p_n = calculsAccAngu(\alpha_n, \beta_n, \gamma_n, \dot{\alpha}_n, \dot{\beta}_n, \dot{\gamma}_n)$$

$$q_n = calculsAccAngu(\alpha_n + \dot{\alpha}_n \cdot \tfrac{h}{2}, \beta_n + \dot{\beta}_n \cdot \tfrac{h}{2}, \gamma_n + \dot{\gamma}_n \cdot \tfrac{h}{2}, \dot{\alpha}_n + p_{n;\alpha} \cdot \tfrac{h}{2}, \dot{\beta}_n + p_{n;\beta} \cdot \tfrac{h}{2}, \dot{\gamma}_n + p_{n;\gamma} \cdot \tfrac{h}{2})$$

$$r_n = calculsAccAngu(\alpha_n \ + \ \dot{\alpha}_n \cdot \tfrac{h}{2} \ + \ \tfrac{1}{2} \cdot p_{n;\alpha} \cdot (\tfrac{h}{2})^2, \beta_n \ + \ \dot{\beta}_n \cdot \tfrac{h}{2} \ + \ \tfrac{1}{2} \cdot p_{n;\beta} \cdot (\tfrac{h}{2})^2, \gamma_n \ + \ \dot{\gamma}_n \cdot \tfrac{h}{2} \ + \ \tfrac{1}{2} \cdot p_{n;\gamma} \cdot (\tfrac{h}{2})^2, \dot{\alpha}_n \ + \ q_{n;\alpha} \cdot \tfrac{h}{2}, \dot{\beta}_n \ + \ q_{n;\beta} \cdot \tfrac{h}{2}, \dot{\gamma}_n \ + \ q_{n;\gamma} \cdot \tfrac{h}{2})$$

$$s_n = calculsAccAngu(\alpha_n \ + \ \dot{\alpha}_n \cdot h \ + \ \tfrac{1}{2} \cdot q_{n;\alpha} \cdot h^2, \beta_n \ + \ \dot{\beta}_n \cdot h \ + \ \tfrac{1}{2} \cdot q_{n;\beta} \cdot h^2, \gamma_n \ + \ \dot{\gamma}_n \cdot h \ + \ \tfrac{1}{2} \cdot q_{n;\gamma} \cdot h^2, \dot{\alpha}_n \ + \ r_{n;\alpha} \cdot h, \dot{\beta}_n \ + \ r_{n;\beta} \cdot h, \dot{\gamma}_n \ + \ r_{n;\gamma} \cdot h)$$

$$\dot{\alpha}_{n+1} = \dot{\alpha}_n \ + \ \frac{p_{n;\alpha} \ + \ 2q_{n;\alpha} \ + \ 2r_{n;\alpha} \ + \ s_{n;\alpha}}{6} \cdot h$$

$$\dot{\beta}_{n+1} = \dot{\beta}_n \ + \ \frac{p_{n;\beta} \ + \ 2q_{n;\beta} \ + \ 2r_{n;\beta} \ + \ s_{n;\beta}}{6} \cdot h$$

$$\dot{\gamma}_{n+1} = \dot{\gamma}_n \ + \ \frac{p_{n;\gamma} \ + \ 2q_{n;\gamma} \ + \ 2r_{n;\gamma} \ + \ s_{n;\gamma}}{6} \cdot h$$

$$\alpha_{n+1} = \alpha_n \ + \ \dot{\alpha}_n \cdot h \ + \ \frac{1}{2} \cdot \frac{p_{n;\alpha} \ + \ q_{n;\alpha} \ + \ r_{n;\alpha}}{3} \cdot h^2$$

$$\beta_{n+1} = \beta_n \ + \ \dot{\beta}_n \cdot h \ + \ \frac{1}{2} \cdot \frac{p_{n;\beta} \ + \ q_{n;\beta} \ + \ r_{n;\beta}}{3} \cdot h^2$$

$$\gamma_{n+1} = \gamma_n \ + \ \dot{\gamma}_n \cdot h \ + \ \frac{1}{2} \cdot \frac{p_{n;\gamma} \ + \ q_{n;\gamma} \ + \ r_{n;\gamma}}{3} \cdot h^2$$

A.3.2 Trajectoire balistique du projectile

Méthode d'Euler

$$t_{n+1} = t_n \ + \ h$$
$$p_n = a(v_{n;x}, v_{n;y})$$
$$v_{n+1;x} = v_{n;x} \ + \ p_{n;x} \cdot h$$
$$v_{n+1;y} = v_{n;y} \ + \ p_{n;y} \cdot h$$
$$x_{n+1} = x_n \ + \ v_{n;x} \cdot h \ + \ \frac{1}{2} \cdot p_{n;x} \cdot h^2$$
$$y_{n+1} = y_n \ + \ v_{n;y} \cdot h \ + \ \frac{1}{2} \cdot p_{n;y} \cdot h^2$$

Méthode de Heun

$$t_{n+1} = t_n \ + \ h$$
$$p_n = a(v_{n;x}, v_{n;y})$$
$$q_n = a(v_{n;x} \ + \ p_{n;x} \cdot h, v_{n;y} \ + \ p_{n;y} \cdot h)$$
$$v_{n+1;x} = v_{n;x} \ + \ \frac{p_{n;x} \ + \ q_{n;x}}{2} \cdot h$$
$$v_{n+1;y} = v_{n;y} \ + \ \frac{p_{n;y} \ + \ q_{n;y}}{2} \cdot h$$
$$x_{n+1} = x_n \ + \ v_{n;x}h \ + \ \frac{1}{2} \cdot \frac{p_{n;x} \ + \ q_{n;x}}{2} \cdot h^2$$
$$y_{n+1} = y_n \ + \ v_{n;y}h \ + \ \frac{1}{2} \cdot \frac{p_{n;y} \ + \ q_{n;y}}{2} \cdot h^2$$

Méthode de Runge - Kutta (ordre 4)

$$t_{n+1} = t_n \ + \ h$$
$$p_n = a(v_{n;x}, v_{n;y})$$
$$q_n = a(v_{n;x} \ + \ p_{n;x} \cdot \frac{h}{2}, v_{n;y} \ + \ p_{n;y} \cdot \frac{h}{2})$$
$$r_n = a(v_{n;x} \ + \ q_{n;x} \cdot \frac{h}{2}, v_{n;y} \ + \ q_{n;y} \cdot \frac{h}{2})$$
$$s_n = a(v_{n;x} \ + \ r_{n;x} \cdot h, v_{n;y} \ + \ r_{n;y} \cdot h)$$
$$v_{n+1;x} = v_{n;x} \ + \ \frac{p_{n;x} \ + \ 2q_{n;x} \ + \ 2r_{n;x} \ + \ s_{n;x}}{6} \cdot h$$
$$v_{n+1;y} = v_{n;y} \ + \ \frac{p_{n;y} \ + \ 2q_{n;y} \ + \ 2r_{n;y} \ + \ s_{n;y}}{6} \cdot h$$
$$x_{n+1} = x_n \ + \ v_{n;x}h \ + \ \frac{1}{2} \cdot \frac{p_{n;x} \ + \ q_{n;x} \ + \ r_{n;x}}{3} \cdot h^2$$
$$y_{n+1} = y_n \ + \ v_{n;y}h \ + \ \frac{1}{2} \cdot \frac{p_{n;y} \ + \ q_{n;y} \ + \ r_{n;y}}{3} \cdot h^2$$

A.4 Perte d'énergie par collisions

Nous nous intéressons en premier lieu au cas dans lequel la fronde rebondit contre le bras principal du trébuchet. La partie de l'énergie qui se verrait modifiée serait l'énergie cinétique de la masse m_2, c'est-à-dire T_2. Le calcul de T_2 est en effet le seul à dépendre de la vitesse angulaire $\dot{\beta}$. Nous calculons donc la différence d'énergies entre l'ancienne énergie T_2 et la nouvelle énergie T_2' :

$$\mathrm{T}_{2;n} = \tfrac{1}{2} m_2 \left[\left({l_2}^2 \ + \ {l_5}^2 \ - \ 2 l_2 l_5 \cdot \cos\left(\beta_n\right) \right) \dot{\alpha}_n^2 \ + \ {l_5}^2 \dot{\beta}_n^2 \ + \ \left(-2 {l_5}^2 \ + \ 2 l_2 l_5 \cos\left(\beta_n\right) \right) \dot{\alpha}_n \dot{\beta}_n \right]$$

$$T'_{2;n} = \tfrac{1}{2} m_2 \left[\left({l_2}^2 + {l_5}^2 - 2 l_2 l_5 \cdot \cos\left(\beta_n\right) \right) \dot{\alpha}_n^2 + {l_5}^2 \left(-e \cdot \dot{\beta}_n \right)^2 + \left(-2 {l_5}^2 + 2 l_2 l_5 \cos\left(\beta_n\right) \right) \dot{\alpha}_n \left(-e \cdot \dot{\beta}_n \right) \right] =$$
$$\tfrac{1}{2} m_2 \left[\left({l_2}^2 \ + \ {l_5}^2 \ - \ 2 l_2 l_5 \cdot \cos\left(\beta_n\right) \right) \dot{\alpha}_n^2 \ + \ {l_5}^2 e^2 \dot{\beta}_n^2 \ - \ e \left(-2 {l_5}^2 + 2 l_2 l_5 \cos\left(\beta_n\right) \right) \dot{\alpha}_n \dot{\beta}_n \right]$$

$$T_{2;n} \ - \ T'_{2;n} = \tfrac{1}{2} m_2 \left[{l_5}^2 \dot{\beta}_n^2 (1 - e^2) \ + \ (1 + e) \left(-2 {l_5}^2 + 2 l_2 l_5 \cos\left(\beta_n\right) \right) \dot{\alpha}_n \dot{\beta}_n \right]$$

Nous faisons de même maintenant avec l'énergie T_1 pour la collision entre le bras et la longueur du contrepoids :

$$T_{1;n} = \tfrac{1}{2} m_1 \left[\left({l_1}^2 \ + \ {l_4}^2 \ - \ 2 l_1 l_4 \cdot \cos\left(\gamma_n\right) \right) \dot{\alpha}_n^2 \ + \ {l_4}^2 \dot{\gamma}_n^2 \ + \ \left(2 l_4^2 \ - \ 2 l_1 l_4 \cos\left(\gamma_n\right) \right) \dot{\alpha}_n \dot{\gamma}_n \right]$$

$$T'_{1;n} = \tfrac{1}{2} m_1 \left[\left({l_1}^2 + {l_4}^2 - 2 l_1 l_4 \cdot \cos\left(\gamma_n\right) \right) \dot{\alpha}_n^2 + {l_4}^2 \left(-e \cdot \dot{\gamma}_n \right)^2 \ + \ \left(2 {l_4}^2 \ - \ 2 l_1 l_4 \cos\left(\gamma_n\right) \right) \dot{\alpha}_n \left(-e \cdot \dot{\gamma}_n \right) \right] =$$
$$\tfrac{1}{2} m_1 \left[\left({l_1}^2 \ + \ {l_4}^2 \ - \ 2 l_1 l_4 \cdot \cos\left(\gamma_n\right) \right) \dot{\alpha}_n^2 \ + \ {l_4}^2 e^2 \dot{\gamma}_n^2 \ - \ e \left(2 {l_4}^2 \ - \ 2 l_1 l_4 \cos\left(\gamma_n\right) \right) \dot{\alpha}_n \dot{\gamma}_n \right]$$

$$T_{1;n} \ - \ T'_{1;n} = \tfrac{1}{2} m_1 \left[{l_4}^2 \dot{\gamma}_n^2 (1 - e^2) + (1 + e) \left(2 {l_4}^2 \ - \ 2 l_1 l_4 \cos\left(\gamma_n\right) \right) \dot{\alpha}_n \dot{\gamma}_n \right]$$

Dès lors la perte maximale d'énergie est de :

$$\Delta E_{max} = \tfrac{1}{2} m_1 \left[{l_4}^2 \dot{\gamma}_n^2 (1 - e^2) \ + \ (1 + e) \left(2 {l_4}^2 - 2 l_1 l_4 \cos\left(\gamma_n\right) \right) \dot{\alpha}_n \dot{\gamma}_n \right]$$
$$+ \tfrac{1}{2} m_2 \left[{l_5}^2 \dot{\beta}_n^2 (1 - e^2) \ + \ (1 + e) \left(-2 {l_5}^2 + 2 l_2 l_5 \cos\left(\beta_n\right) \right) \dot{\alpha}_n \dot{\beta}_n \right]$$